Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM 2001), Pages 521--528, California, USA, November 2001.

Hierarchical Text Classification and Evaluation

Aixin Sun and Ee-Peng Lim
Center for Advanced Information Systems
Nanyang Technological University
Nanyang Avenue, Singapore 639798, Singapore
sunaixin@pmail.ntu.edu.sg aseplim@ntu.edu.sg

Abstract

Hierarchical Classification refers to assigning of one or more suitable categories from a hierarchical category space to a document. While previous work in hierarchical classification focused on virtual category trees where documents are assigned only to the leaf categories, we propose a top-down level-based classification method that can classify documents to both leaf and internal categories. As the standard performance measures assume independence between categories, they have not considered the documents incorrectly classified into categories that are similar or not far from the correct ones in the category tree. We therefore propose the Category-Similarity Measures and Distance-Based Measures to consider the degree of misclassification in measuring the classification performance. An experiment has been carried out to measure the performance of our proposed hierarchical classification method. The results showed that our method performs well for Reuters text collection when enough training documents are given and the new measures have indeed considered the contributions of misclassified documents.

1. Introduction

Text classification (TC) or text categorization is the process of automatically assigning one or more predefined categories to text documents. In TC research, most of the studies have focused on *flat classification* where the predefined categories are treated in isolation and there is no structure defining the relationships among them [1, 19]. Such categories are also known as *flat categories*. However, when the number of categories grows to a significantly large number, it will become much more difficult to browse and search the categories. One way to solve this problem is to organize the categories into a hierarchy like the one developed by Yahoo! [18].

Hierarchical classification allows us to address a large classification problem using a divide-and-conquer approach. At the root level in the category hierarchy, a document can be first classified into one or more sub-categories using some flat classification method(s). The classification can be repeated on the document in each of the sub-categories until the document reaches some leaf categories or cannot be further classified into any sub-categories. A few hierarchical classification methods have been proposed recently [1, 2, 10, 13, 15, 17]. In most of the hierarchical classification methods, the categories are organized in tree-like structures. On the whole, we can identify four distinct category structures for text classification. They are:

1. *Virtual category tree:* In this category structure, categories are organized as a tree. Each category can belong to at most one parent category and documents can *only* be assigned to the leaf categories [2].
2. *Category tree:* This is an extension of the virtual category tree that allows documents to be assigned into both internal and leaf categories [15].
3. *Virtual directed acyclic category graph:* In this category structure, categories are organized as a Directed Acyclic Graph (DAG). Similar to the virtual category tree, documents can only be assigned to leaf categories.
4. *Directed acyclic category graph:* This is perhaps the most commonly-used structure in the popular web directory services such as Yahoo! [18] and Open Directory Project [11]. Documents can be assigned to both internal and leaf categories.

In this paper, we will only focus on hierarchical classification that involves category trees.

To compare different hierarchical classification methods, experiments involving training and test data sets have to be conducted, and some performance measures are used to determine the effectiveness of the methods. In flat classification, performance measures such as *precision* and *recall* have been widely used [14, 19]. The same performance

measures have also been used to measure the performance of hierarchical classification methods. In this paper, we argue that these performance measures are not adequate as they have largely ignored the parent-child and sibling relationships between categories in a hierarchy. By not considering the "closeness" of categories, the performance of hierarchical classification may not be accurately captured. In general, the categories from the same subtree share more domain knowledge than the ones from different subtrees, that is, the categories from the same subtree are semantically closer to one another. With the standard precision and recall measures, all these relationships among categories are not accounted for. In this paper, we will present several performance measures applicable to hierarchical classification. Among them are the *category similarity measures* and *distance-based measures*.

In this paper, we aim to establish a framework to evaluate the performance of hierarchical classification. There are two main contributions:

1. We define a new set of performance measures that consider the semantic relationships and parent-child relationships among categories in a hierarchy. The intuition is that when a document is wrongly classified, one has to examine how different is the incorrect category from the correct category.

2. We develop a top-down level-based hierarchical classification method for category tree using Support Vector Machine (SVM) classifiers. By conducting experiments using the Reuters text collection and the new performance measures, we illustrate how the performance of hierarchical classification can be more accurately determined.

This paper is organized as follows. We first give an overview of the related hierarchical classification work in Section 2. In Section 3, we present several new performance measures for hierarchical classification. The experiment on our proposed hierarchical classification method will be described in Section 5. The results of both standard performance measures and the new performance measures are presented in this section. Finally, we conclude our work in Section 6.

2. Related work

The existing hierarchical classification methods have mostly assumed a *virtual category tree* structure [2, 13]. Furthermore, these methods have often been evaluated using the performance measures developed for flat classification. There are basically two approaches adopted by the existing hierarchical classification methods, namely, the *big-bang approach* and the *top-down level-based approach*.

In the big-bang approach, only a single classifier is used in the classification process. Given a document, the classifier assigns it to one or more categories in the category tree. The assigned categories can be internal or leaf categories depending on the category structure supported by the methods. The big-bang approach has been achieved with Rocchio-like classifier [7], rule-based classifier [13] and methods built upon association rule mining [15]. The performance measures used in these experiments have been very much based on simple empirical observations of the number of correctly classified documents or the percentage of incorrectly classified documents.

In the top-down level-based approach, one or more classifiers are constructed at each level of the category tree and each classifier works as a flat classifier at that level. A document will first be classified by the classifier at the root level into one or more lower level categories. It will then be further classified by the classifier(s) of the lower level category(ies) until it reaches a final category which could be a leaf category or an internal category. The top-down level-based classification has been implemented with ACTION (for Automatic Classification for Full-Text Documents) algorithm in [1], multiple Bayesian classifiers in [6] and Support Vector Machine classifiers in [2]. Three performance measures, i.e., precision, recall and F-measure have been used in these experiments.

Compared to the top-down level-based approach, the big-bang approach can only use the information carried by the category structure during the training phase but not the classification phase. As discriminative features (e.g., terms) at a parent category may not be discriminative at the child categories, it is usually very difficult for a classification method using big-bang approach to exploit different sets of features at different category levels. Another issue in the big-bang approach is that the classifier constructed may not be flexible enough to cater for changes to the category structure. The classifier needs to be retrained once the category structure is changed.

On the other hand, the top-down level-based classification approach is not problem-free. One of its obvious problems is that a misclassification at a parent (ancestor) category may force a document to be excluded from the child categories before it could be examined by the classifiers of the child categories. Classification methods based on top-down approach also require more training examples since multiple classifiers have to be constructed and each requires a different training set. Without adequate training examples, the performance of these classifiers may suffer.

3. Performance measures

To evaluate a hierarchical classification method, one can directly apply the standard precision and recall for flat clas-

Category C_i		Expert Judgments	
		YES	NO
Classifier Judgments	YES	TP_i	FP_i
	NO	FN_i	TN_i

Table 1. Contingency table for category C_i

sification on each category of the entire category space.

Most hierarchical classification methods that involve virtual category trees often exclude the internal categories from the performance measurement as they are virtual categories and there are no documents under them. In a category tree, however, all categories have to be considered. In addition, categories are connected with *parent-child* and *sibling* relationships. Two categories can be similar when they share many common documents. For example, the *Programming* and *Software Engineering* categories may have several common features allowing documents to be classified under both of them. Therefore, if a document is not classified to the correct category, one should consider the degree of wrong classification. In other words, wrongly classifying a document into a parent or child category is considered better than classifying it into categories that are far away from the correct category.

3.1. Measures for flat classification

The performance of text classification methods can be measured in several ways. In this paper, we are interested in measuring the accuracy of the final classification results, i.e., the correctness of assigning categories to a set of documents. According to the text classification survey by Sebastiani [14], the most commonly used performance measures in flat classification are the classic information retrieval (IR) notions of *Precision* and *Recall*. Precision for a category C_i, denoted as Pr_i, measures the percentage of correct assignments among all the documents assigned to C_i. The Recall Re_i gives the percentage of correct assignments in C_i among all the documents that should be assigned to C_i. Pr_i and Re_i are also known as the standard precision and recall for C_i in this paper. The contingency table for a particular category C_i from the category space $\{C_1, \ldots, C_m\}$ is shown in Table 1. Let TP_i be the set of documents correctly classified into category C_i; FP_i be the set of documents wrongly classified; FN_i be the set of documents wrongly rejected and TN_i be the set of documents correctly rejected. The standard precision and recall are defined as follows[1]:

$$Pr_i = \frac{|TP_i|}{|TP_i| + |FP_i|} \quad (1)$$

[1]In all the formulas presented in this paper, $|S|$ gives the number of elements in set S.

$$Re_i = \frac{|TP_i|}{|TP_i| + |FN_i|} \quad (2)$$

Based on the standard precision and recall for each category, the overall precision and recall for the whole category space, i.e., $\{C_1, \ldots, C_m\}$, can be obtained in two ways, namely, *Micro-Average* and *Macro-Average*. Micro-Average gives equal importance to each document, while Macro-Average gives equal importance to each category [19]:

1. *Micro-Average*:

$$\hat{Pr}^{\mu} = \frac{\sum_{i=1}^{m} |TP_i|}{\sum_{i=1}^{m} (|TP_i| + |FP_i|)} \quad (3)$$

$$\hat{Re}^{\mu} = \frac{\sum_{i=1}^{m} |TP_i|}{\sum_{i=1}^{m} (|TP_i| + |FN_i|)} \quad (4)$$

2. *Macro-Average*:

$$\hat{Pr}^{M} = \frac{\sum_{i=1}^{m} Pr_i}{m} \quad (5)$$

$$\hat{Re}^{M} = \frac{\sum_{i=1}^{m} Re_r}{m} \quad (6)$$

Neither precision nor recall can effectively measure classification performance in isolation [14]. Therefore, the performance of the text classification has often been measured by the combination of the two measures. The popular combinations are listed below:

1. *Break-Even Point*(BEP): BEP, proposed by Lewis [8], defines the point at which precision and recall are equal. However, in some cases, BEP can never be obtained. For example, if there are only a few positive test documents compared to a large number of negative ones, the recall value can be so high that the precision can never reach.

2. F_β *Measure*: F_β measure was proposed by Rijsbergen [12]. It is a single score computed from precision and recall values according to the user-defined importance (i.e., β) of precision and recall. Normally, $\beta = 1$ is used [19]. The formula is:

$$F_\beta = \frac{(\beta^2 + 1) \cdot P_r \cdot R_e}{\beta^2 \cdot P_r + R_e} \quad \text{where } \beta \in [0, \infty) \quad (7)$$

3. *Average 11-Point Precision*: The precision values are interpolated at 11 points at which the recall values are 0.0, 0.1, …, 1.0. This measure is mostly used in the situation when the classification method ranks documents according to their appropriateness to a category or similarly ranks categories to a document [14].

Besides precision and recall, other commonly-used performance measures include *Accuracy* and *Error* [14, 6], denoted by Ac_i and Er_i for category C_i respectively.

$$Ac_i = \frac{|TP_i| + |TN_i|}{|TP_i| + |TN_i| + |FP_i| + |FN_i|} \quad (8)$$

$$Er_i = \frac{|FP_i| + |FN_i|}{|TP_i| + |TN_i| + |FP_i| + |FN_i|} = 1 - Ac_i \quad (9)$$

3.2. Measures based on category similarity

Intuitively, if a classification method A misclassifies documents into categories similar to the correct categories, it is considered better than another method, say B, that misclassifies the documents into totally unrelated categories. We therefore extend the standard precision and recall definitions to distinguish the performance of A and B.

The *Category Similarity* between two categories C_i and C_k, denoted by $CS(C_i, C_k)$, can be computed in several ways. In our work, we have chosen to adopt cosine distance between the feature vectors of two categories. It is suggested that the feature vector for a category should be derived by summation of the feature vectors of all training documents under it. The feature vectors of documents are the ones used to build the classifiers. From the category similarities, one can define the *Average Category Similarity* (*ACS*). The formulas for CS and ACS are:

$$C_i = \{w_1 t_1, w_2 t_2, \ldots, w_N t_N\}$$

$$C_k = \{v_1 t_1, v_2 t_2, \ldots, v_N t_N\}$$

$$CS(C_i, C_k) = \frac{\sum_{n=1}^{N}(w_n \times v_n)}{\sqrt{\sum_{n=1}^{N} w_n^2 \times \sum_{n=1}^{N} v_n^2}} \quad (10)$$

$$ACS = \frac{2 \times \sum_{i=1}^{m} \sum_{k=i+1}^{m} CS(C_i, C_k)}{m \times (m-1)} \quad (11)$$

where t_n's are index terms, w_n's and v_n's are the corresponding term weights.

Based on the category similarity, we can now measure the degree of correctness of the assigned categories $d_j.agd$ of document d_j while its labelled categories are $d_j.lbd$. In the simplest case where d_j is assigned to C_i correctly, i.e., $d_j \in TP_i$, d_j is counted as 1 in computation of precision and recall for C_i, similar to flat classification. However, if d_j is wrongly assigned to C_i (i.e., $d_j \in FP_i$), we should consider whether the d_j's labelled categories are similar to C_i; that is, how much d_j can *contribute* to C_i when we compute the precision and recall values for C_i. In this case, the $contribution$ of d_j to C_i, denoted by $Con(d_j, C_i)$ is defined as follows:

$$Con(d_j, C_i) = \frac{\sum_{C' \in d_j.lbd}(CS(C', C_i) - ACS)}{1 - ACS} \quad (12)$$

Similarly, if d_j is wrongly rejected from C_i, say, $d_j \in FN_i$ the $contribution$ of d_j to C_i depends on the category similarities between C_i and the assigned categories of d_j.

$$Con(d_j, C_i) = \frac{\sum_{C' \in d_j.agd}(CS(C', C_i) - ACS)}{1 - ACS} \quad (13)$$

The contribution of a document can be positive or negative depending on how similar its labelled and assigned categories are in comparison with the average category similarity ACS. Note that a document can belong to or be assigned to more than one category. To prevent one document from being over-shined or over-punished, the $contribution$ of each document d_j to category C_i should be restricted to the range of $[-1, 1]$. Therefore, the $Refined-Contribution$, denoted by $RCon(d_j, C_i)$ is defined as follows:

$$RCon(d_j, C_i) = \min(1, \max(-1, Con(d_j, C_i))) \quad (14)$$

For all the documents that belong to FP_i, the total contributions $FpCon_i$ will be:

$$FpCon_i = \sum_{d_j \in FP_i} RCon(d_j, C_i) \quad (15)$$

and similarly, the total contributions $FnCon_i$ is:

$$FnCon_i = \sum_{d_j \in FN_i} RCon(d_j, C_i) \quad (16)$$

The extended $Precision$ Pr_i^{CS} and $Recall$ Re_i^{CS} for category C_i based on category similarity are defined as follows:

$$Pr_i^{CS} = \frac{\max(0, |TP_i| + FpCon_i + FnCon_i)}{|TP_i| + |FP_i| + FnCon_i} \quad (17)$$

$$Re_i^{CS} = \frac{\max(0, |TP_i| + FpCon_i + FnCon_i)}{|TP_i| + |FN_i| + FpCon_i} \quad (18)$$

Since both $FpCon_i$ and $FnCon_i$ can be negative, $|TP_i| + FpCon_i + FnCon_i$ can be negative. Therefore, a max function is applied to the numerator to make it not less than 0. As $FpCon_i \leq |FP_i|$, when $|TP_i| + |FP_i| + FnCon_i \leq 0$, the numerator $max(0, |TP_i| + FpCon_i + FnCon_i) = 0$ and Pr_i^{CS} can be treated as 0 in this case. The same rule is applicable to Re_i^{CS}.

The *Micro-Average* and *Macro-Average* can be extended to consider category similarity. We give the extended definitions as follows:

Micro-Average:

$$\hat{Pr}^{\mu_{CS}} = \frac{\sum_{i=1}^{m}(\max(0, |TP_i| + FpCon_i + FnCon_i))}{\sum_{i=1}^{m}(|TP_i| + |FP_i| + FnCon_i)} \quad (19)$$

$$\hat{Re}^{\mu_{CS}} = \frac{\sum_{i=1}^{m}(\max(0, |TP_i| + FpCon_i + FnCon_i))}{\sum_{i=1}^{m}(|TP_i| + |FN_i| + FpCon_i)} \quad (20)$$

Macro-Average:

$$\hat{Pr}^{M_{CS}} = \frac{\sum_{i=1}^{m} Pr_i^{CS}}{m} \quad (21)$$

$$\hat{Re}^{M_{CS}} = \frac{\sum_{i=1}^{m} Re_r^{CS}}{m} \quad (22)$$

Similar to the extended precision and recall, the extended *accuracy* and *error* for category C_i can be defined based on document contribution:

$$Ac_i^{CS} = \frac{|TP_i| + |TN_i| + FpCon_i + FnCon_i}{|TP_i| + |TN_i| + |FP_i| + |FN_i|} \quad (23)$$

$$Er_i^{CS} = \frac{|FP_i| + |FN_i| - FpCon_i - FnCon_i}{|TP_i| + |TN_i| + |FP_i| + |FN_i|} \quad (24)$$

Note that the sum of extended accuracy and error is 1 which is the same as the original definitions.

As we have discussed in Section 3.1, it is not sufficient to evaluate a classification method using only precision or recall. Instead, they have to be considered together. The performance measures that combine both precision and recall are the Break-Even Point (BEP), F_β and Average 11-Point Precision. Among them, F_β can be easily computed using the extended precision and recall. BEP can be applied to classification methods that can rank documents for each category. In hierarchical classification using the big-bang approach, BEP and Average 11-Point Precision can be computed for classification methods that can rank *all* documents in the test set for each category. On the other hand, for those classification methods using top-down level-based approach, the test documents available for classification at a level are determined by the parent classifier as the latter may reject documents before they reach the child classifier(s). With such restriction, it is difficult to compute the BEP and Average 11-Point Precision for each category. Hence, we argue that the above two performance measures are less applicable to the hierarchical classification methods.

3.3. Measures based on category distance

Instead of using category similarity, we can define performance measures based on the distances between categories in a category tree. The distance between two categories C_i and C_k, denoted by $Dis(C_i, C_k)$, is defined to be the number of the links between C_i and C_k. Intuitively, the shorter the length, the closer the two categories.

The distance between categories was first proposed to measure misclassification in [16]. Nevertheless, the work did not define performance measures based on category distance. To define the $contribution$ of misclassified documents, an *acceptable distance*, denoted as Dis_θ, must first be specified by the user. Dis_θ must be greater than 0. For example, if $Dis_\theta = 1$, a misclassification of document that involves the labelled and assigned categories at more than 1 link apart will yield negative contribution, but zero contribution at 1 link apart. Formally, the $contribution$ of a document d_j to category C_i based on category distance is defined as follows:

- If $d_j \in FP_i$:

$$Con(d_j, C_i) = \sum_{C' \in d_j.lbd} (1.0 - \frac{Dis(C', Ci)}{Dis_\theta}) \quad (25)$$

- If $d_j \in FN_i$:

$$Con(d_j, C_i) = \sum_{C' \in d_j.agd} (1.0 - \frac{Dis(C', Ci)}{Dis_\theta}) \quad (26)$$

For the same reason, the $contribution$ needs to be refined to be in the range of $[-1, 1]$. With this new definition for *contribution*, the extended precision and recall based on category distance, denoted by Pr_i^{DB} and Re_i^{DB}, can be defined using the formulae (17) and (18) respectively. Similarly, *Micro-Average*, *Macro-Average*, F_β, *accuracy* and *error* can be extended.

4. Hierarchical classification method

In this section, we propose a hierarchical classification method for *category tree* structure based on top-down level-based approach. All the classifiers involved in this method are binary classifiers. Binary classifiers normally need to be trained with both positive and negative training documents. In the hierarchical classification method, a binary classifier is built for each category. These classifiers that determine whether a document should belong to the corresponding categories are known as the *local-classifiers*. However, an additional binary classifier is built for each internal category to determine whether a document should be given to the classifiers of its sub-categories. This special classifier is known as the *subtree-classifier* since it decides whether a document should belong to a subtree. This separation of local and subtree classifiers distinguishes our method from that proposed by Dumais and Chen [2]. To build binary classifiers in hierarchical classification, special consideration must be given to the selection of training documents for each classifier.

The $Coverage$ of a category C_i in a given category tree, denoted by $Coverage(C_i)$ is the set of categories that belongs to the subtree rooted at C_i including C_i. For example, in the hierarchy shown in Figure 1 Tree (a), $Coverage(grain) = \{wheat, corn, grain\}$. For any document d_j, $d_j \in C_i$ is true if and only if d_j belongs to category C_i; $d_j \in Coverage(C_i)$ is true if and only if d_j belongs to any of the categories in $Coverage(C_i)$.

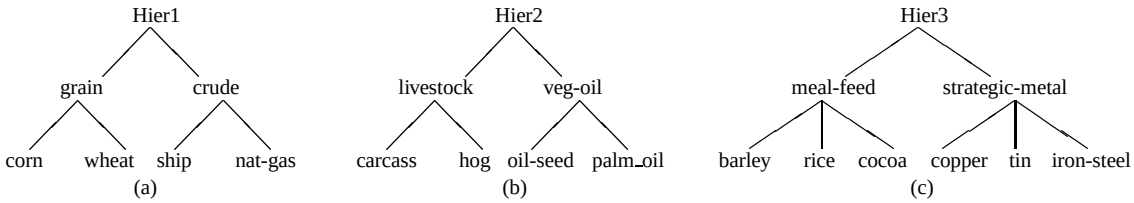

Figure 1. Category trees from Reuters collection

$Parent(C_i)$ returns the parent category of category C_i. Let $\{d_1, \ldots, d_m\}$ denote a set of training documents. The positive and negative training documents are selected differently for each type of classifier given below[2] where *+ve* and *-ve* refer to positive and negative training documents respectively and $d_j \in \{d_1, \ldots, d_m\}$.

- Subtree-classifier of a root category C_{root}:
  - +ve: All d_j such that $d_j \in Coverage(C_{root})$.
  - -ve: All d_j or Equal number (to positive documents) of random selected d_j such that $d_j \notin Coverage(C_{root})$. The number of d_j selected and the method of selection are classification methods dependent for this subtree-classifier.
- Subtree-classifier of an internal category C_i:
  - +ve: All d_j such that $d_j \in Coverage(C_i)$.
  - -ve: All d_j such that $d_j \notin Coverage(C_i)$ and $d_j \in Coverage(Parent(C_i))$.
- Local-classifier of an internal category C_i:
  - +ve: All d_j such that $d_j \in C_i$.
  - -ve: All d_j such that $d_j \notin C_i$ and $d_j \in Coverage(C_i)$.
- Local-classifier of a leaf category C_l:
  - +ve: All d_j such that $d_j \in C_l$.
  - -ve: All d_j such that $d_j \notin C_l$ and $d_j \in Coverage(Parent(C_l))$.

5. Experiments with SVM classifier

Support Vector Machine (SVM) classifiers have been shown to be fast and effective in text classification [3, 5]. Dumais and Chen have shown that SVM works well for virtual category tree but they did not consider category tree in their work [2]. The purpose of this experiment is to explore the use of SVM classifiers in classifying documents into a category tree. In the experiments, we compared the classification performance indicated by both the standard and extended precision and recall. The SVM classifier used in our experiment is *SVM* ^{light} Version 3.50 implemented by Joachims [4].

In our experiment, Reuters-21578 collection[3] was used. To conduct our experiment, category trees need to be manually derived from the 135 categories. Kohler and Sahami extracted three category trees from the Reuters-22173 collection by identifying the category labels that suggest parent-child relationships [6]. Three slightly different category trees are derived from the Reuters-21578 collection using a similar approach (see Figure 1). Note that the roots of the three category trees are virtual categories.

Almost all documents in Reuters collection come with title, dateline and text body. We obtained the index terms from only the title and text body after stopword removal and stemming. The stopwords and the stemming algorithms have been taken directly from the BOW library [9]. A binary term vector is obtained for each document without applying any feature selection. In our experiment, we used *Lewis Split* provided by Reuters collection to obtain training documents and test documents.

All the test documents that belong to one or more category in a category tree are used as its positive test documents. The same number of test documents that do not belong to the category tree are randomly selected to be the negative test documents. The statistics about our training and test documents are shown in Table 2. For Category C_i, C_i:s and C_i:l refer to the subtree-classifier and local-classifier for C_i respectively. In the table, $+Tr$, $-Tr$ and $+Te$ refer to number of positive training, negative training and positive test documents respectively. The category similarity matrix for Tree(a) is shown in Table 3. As $CS(C_i, C_k) = CS(C_k, C_i)$, only the lower half of the matrix is shown. The ones for Tree(b) and Tree (c) can be computed in a similar way.

The results of our experiment are shown in Tables 4, 5 and 6 for the three category trees. We computed the standard *Precision* Pr and *Recall* Re, the precision Pr^{CS} and recall Re^{CS} based on category similarity; and the precision Pr^{DB} and recall Re^{DB} based on category distance for

[2] As normally there is no document directly under the root category, the local-classifier for the root category is not constructed.

[3] http://www.research.att.com/~lewis/reuters21578.html

Tree (a)				Tree (b)				Tree (c)			
Category	$+Tr$	$-Tr$	$+Te$	Category	$+Tr$	$-Tr$	$+Te$	Category	$+Tr$	$-Tr$	$+Te$
Hier1	981	981	394	Hier2	270	270	104	Hier3	271	271	123
crude:s	574	435	-	livestock:s	83	188	-	meal-feed:s	153	119	-
crude:l	391	183	189	livestock:l	75	8	24	meal-feed:l	30	123	19
grain:s	437	544	-	veg-oil:s	191	81	-	str-metal:s	119	153	-
grain:l	434	3	149	veg-oil:l	87	104	37	str-metal:l	16	103	11
nat-gas	75	499	30	carcass	50	33	18	barley	37	116	14
ship	198	376	89	hog	16	67	6	rice	55	98	18
corn	182	255	56	oil-seed	124	67	47	cocoa	35	118	24
wheat	212	225	71	palm-oil	30	161	10	copper	47	72	18
-	-	-	-	-	-	-	-	iron-steel	40	79	14
-	-	-	-	-	-	-	-	tin	18	101	12

Table 2. Number of training and test documents for Trees

Category	crude	grain	nat-gas	ship	corn	wheat
crude	1.000	-	-	-	-	-
grain	0.523	1.000	-	-	-	-
nat-gas	0.602	0.453	1.000	-	-	-
ship	0.574	0.556	0.432	1.000	-	-
corn	0.487	0.791	0.463	0.510	1.000	-
wheat	0.497	0.815	0.472	0.513	0.699	1.000
Average Category Similarity			0.559			

Table 3. Category similarity matrix for Tree (a)

Category	Pr	Re	Pr^{CS}	Re^{CS}	Pr^{DB}	Re^{DB}
crude	0.846	0.962	0.849	0.963	0.890	0.964
grain	0.869	0.939	0.869	0.938	0.868	0.932
nat-gas	0.818	0.600	0.833	0.614	0.900	0.714
ship	0.879	0.820	0.879	0.821	0.888	0.843
corn	0.888	0.857	0.944	0.939	0.929	0.913
wheat	0.886	0.986	0.976	0.993	0.942	0.980
Micro-Ave	0.864	0.909	0.883	0.919	0.892	0.922
Macro-Ave	0.864	0.860	0.892	0.878	0.903	0.891

Table 4. Testing results for Tree (a)

each category. The Pr^{DB} and Re^{DB} are computed with $Dis_\theta = 2$. We also computed the Micro-Averages and Macro-Averages for the three definitions of precision and recall.

From Table 2, Tree (a) receives the largest number of training and test documents. Most of the precision and recall (both the standard and extended) values for Tree (a) are good. This is consistent with the experimental results given by Koller and Sahami in [6] although slightly different collections are used. The extended precision and recall based on category similarity are better than the standard ones in most cases (i.e., except the category *grain*). From the category similarity matrix shown in Table 3, we observe that the categories within one subtree are more similar to each other compared to the categories across the two subtrees. In top-down level-based approach, most of the misclassification will occur in the subtrees, unless the document is wrongly rejected at the level *grain* and *crude*. Therefore,

Category	Pr	Re	Pr^{CS}	Re^{CS}	Pr^{DB}	Re^{DB}
livestock	0.629	0.708	0.816	0.771	0.734	0.654
veg-oil	0.878	0.783	0.906	0.845	0.904	0.880
carcass	0.611	0.611	0.778	0.734	0.600	0.473
hog	1.000	0.333	1.000	0.287	1.000	0.416
oil-seed	0.646	0.893	0.688	0.919	0.625	0.901
palm-oil	1.000	0.700	1.000	0.799	1.000	0.918
Micro-Ave	0.710	0.760	0.789	0.815	0.734	0.769
Macro-Ave	0.794	0.671	0.864	0.726	0.810	0.695

Table 5. Testing results for Tree (b)

Category	Pr	Re	Pr^{CS}	Re^{CS}	Pr^{DB}	Re^{DB}
meal-feed	1.000	0.315	1.000	0.332	1.000	0.368
str-metal	0.000	0.000	0.000	0.000	0.000	0.000
barley	0.923	0.857	0.923	0.857	0.923	0.857
rice	1.000	0.833	1.000	0.833	1.000	0.833
cocoa	1.000	0.500	1.000	0.505	1.000	0.520
copper	0.937	0.833	0.937	0.833	0.937	0.833
iron-steel	0.500	0.428	0.500	0.428	0.500	0.428
tin	1.000	0.250	1.000	0.249	1.000	0.166
Micro-Ave	0.896	0.530	0.896	0.534	0.896	0.534
Macro-Ave	0.795	0.502	0.795	0.505	0.795	0.501

Table 6. Testing results for Tree (c)

the documents misclassified within the subtrees should contribute positively to the extended precision and recall based on category similarity. Therefore, it is reasonable for our extended precision and recall to be better than the standard ones.

The same observation holds for the extended precision and recall based on category distance, i.e., most of the extended precision and recall values are higher than the standard ones. Since the acceptable error distance is 2, the documents misclassified within the subtrees contributed positively to the extended precision and recall.

For Trees (b) and (c), there are several categories (i.e., *hog*, *tin* and *strategic-metal*) trained with documents fewer than 20. Since SVM classifiers require about 20 training documents to yield stable performance [2], the performance result may not be representative enough for analysis. Al-

though the extended precision and recall based on either category similarity or category distance give different values compared to the standard precision and recall, it is not clear enough to conclude the performance of our method for Tree(b) and (c).

In summary, our method performed reasonably well for the Reuters collection when given enough training documents. The extended measures have indeed considered contributions of wrongly classified documents.

6. Conclusions

In this paper, we give an overview of hierarchical classification problem and its solutions. While the commonly-used performance measures are the ones for flat classification, the relationships among the categories in category tree are not accounted for. We propose some novel approaches to include the contributions of wrongly classified documents towards performance measures. We have also developed a top-down level-based classification method using binary classifiers (such as SVM) and evaluated the method using the Reuters collection. The results show that our method works well and the extended precision and recall can be feasibly implemented.

In our future work, we are going to evaluate the hierarchical classification method using classifiers other than SVM to compare their performance using the extended measures. Since we use category similarity and distance to measure the classification results, we plan to design a new hierarchical classification method that makes use of such information. In the top-down level-based approach, the error made at the parent category is not recoverable at the child category. We will also try to design a more tolerant hierarchical classification method with which the child classifiers are able to recover the errors made by the parent classifier(s).

References

[1] S. D'Alessio, K. Murray, R. Schiaffino, and A. Kershenbaum. The effect of using hierarchical classifiers in text categorization. In *Proc. of the 6th Int. Conf. "Recherche d'Information Assistee par Ordinateur"*, pages 302–313, Paris, FR, 2000.

[2] S. Dumais and H. Chen. Hierarchical classification of Web content. In *Proc. of the 23rd ACM Int. Conf. on Research and Development in Information Retrieval*, pages 256–263, Athens, GR, 2000.

[3] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proc. of the 7th Int. Conf. on Information and Knowledge Management*, pages 148–155, 1998.

[4] T. Joachims. SVM^{light}, an implementation of Support Vector Machines (SVMs) in C. http://ais.gmd.de/~thorsten/svm_light/.

[5] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In *Proc. of the 10th European Conf. on Machine Learning*, pages 137–142, Chemnitz, DE, 1998.

[6] D. Koller and M. Sahami. Hierarchically classifying documents using very few words. In *Proc. of the 14th Int. Conf. on Machine Learning*, 1997.

[7] Y. Labrou and T. W. Finin. Yahoo! as an ontology: Using Yahoo! categories to describe documents. In *Proc. of the 8th Int. Conf. on Information Knowledge Management*, pages 180–187, Kansas City, MO, 1999.

[8] D. D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In *Proc. of the 15th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 37–50, 1992.

[9] A. K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/∼mccallum/bow, 1996.

[10] D. Mladenic. Turning yahoo to automatic web-page classifier. In *Proc. of the European Conf. on Artificial Intelligence*, pages 473–474, 1998.

[11] ODP - Open Directory Project. http://dmoz.org/.

[12] C. J. V. Rijsbergen. *Information Retrievel*. London: Butterworths, 2nd edition, 1979.

[13] M. Sasaki and K. Kita. Rule-based text categorization using hierarchical categories. In *Proc. of the IEEE Int. Conf. on Systems, Man, and Cybernetics*, pages 2827–2830, La Jolla, US, 1998.

[14] F. Sebastiani. Machine learning in automated text categorisation: a survey. Technical Report IEI-B4-31-1999, Istituto di Elaborazione dell'Informazione, Consiglio Nazionale delle Ricerche, Pisa, IT, 1999. Revised version, 2001.

[15] K. Wang, S. Zhou, and Y. He. Hierarchical classification of real life documents. In *Proc. of the 1st SIAM Int. Conf. on Data Mining*, Chicago, 2001.

[16] K. Wang, S. Zhou, and S. C. Liew. Building hierarchical classifiers using class proximity. In *Proc. of the 25th Int. Conf. on Very Large Data Bases*, pages 363–374, Edinburgh, UK, 1999.

[17] A. S. Weigend, E. D. Wiener, and J. O. Pedersen. Exploiting hierarchy in text categorization. *Information Retrieval*, 1(3):193–216, 1999.

[18] Yahoo! http://www.yahoo.com.

[19] Y. Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1-2):69–90, 1999.