Hierarchical Tree Construction Based on Learning from Classification Results

JuiHsi Fu SingLing Lee

Department of Computer Science and Information Engineering

National Chung Cheng University

168 University Road, Minhsiung Township,

62162 Chiayi, Taiwan, R. O. C.

{fjh95p, singling}@cs.ccu.edu.tw

***Abstract*—Based on the hierarchical taxonomy, a large multi-class classification problem is divided into multiple sub-problems, and the accuracy and efficiency of classification performance could be significantly improved. It is observed that, classification results of validation data could be used for adjusting class nodes in the hierarchical tree, not only for measuring the quality of a hierarchy. If two classes in which documents are frequently misclassified to each are merge, the classification to distinguish between them could be performed more precisely in this new branch. Also, the number of common features of documents between two classes is measured for the similarity of these two classes, and the size of positive training data is concerned for avoiding the overfitting problem in classification. Hence, in this paper, classification results and the number of feature sets and training sets are used for constructing a hierarchical taxonomy. Moreover, three merge-prohibiting conditions are defined for guaranteeing that (1) training sets are consistent with classification models and (2) the overfitting problem is avoided. In our experiments on real-world datasets, it is presented that hierarchical classification accuracy is improved by the proposed method, especially when the classes with few documents are ignored.**

***Index Terms*—Hierarchical Taxonomy Construction, Hierarchical Classification, Document Classification**

This work is supported by NSC, Taiwan, R.O.C. under grant no. NSC 97-2221-E-194-029-MY2.

1. INTRODUCTION

With convenience the Internet brings, a large number of information is easily and frequently published for business or entertainments, such as e-mail, web pages, news, official documents, blogs, etc. However, it becomes a difficult problem in the real world that how to categorize and manage all kinds of electronic text document efficiently. This need motivates experts from varies fields, like linguists, statisticians, and computer scientists to focus on the filed of document classification.

Multi-class document classification is a complicated problem in which there are multiple class labels and each document is labeled by one of them. Also, it is worth noticing that real-world datasets, like News, the catalog of Library, and Wikipedia [1] are all in hierarchical structures. Generally, hierarchical taxonomies are much more intuitive and comprehensible than the flat one. Because the subjects with semantically similar topics are grouped into a category, hierarchical taxonomies could bring convenience for management and categorization potentially. Hence, hierarchical taxonomies are useful tools for multi-class document classification. They could partition a large classification problem into multiple sub-problems to improve the performance of efficiency and accuracy [2], [3], [4], [5]. However, in previous research results, almost all hierarchical taxonomies for different kinds of documents are manually generated. It is inefficient and expensive to define a hierarchical taxonomy by hand in an information system which might need to process millions of documents. Thus, efficient algorithms are needed to be designed for automatically constructing hierarchical taxonomies.

Recently, clustering approaches [6], [7] are frequently used for hierarchical taxonomy constructions. Clustering is a method of unsupervised learning and a common technique for statistical data analysis. Initially, each class is treated as a cluster. Then, similar clusters are merged, and the hierarchy is generated by the structure of merged clusters. However, clustering algorithms are usually limited to the predefined parameters, like the number of clusters or the depth of the child nodes. That is not practical for hierarchical taxonomy construction in real-world applications. In [8], authors use classification performance to adjust the class hierarchy than predefining parameters. At first, the classification of validation data is performed on the predefined hierarchy. Then, location of nodes with bad performance are adjusted, and classification of validation data is performed on the new class hierarchy. This iteration continues until no performance improvement is made by any adjusting.

In this paper, in addition to utilizing performance (accuracy/precision) of the validation data, classification results are also measured for constructing class hierarchy. According to the classification results, it is observed that some documents belonging to a class are frequently misclassified to another class label. If these two classes are merged, the difference among documents belonging to them could be distinguished more clear in this new branch than that in the previous hierarchy. It is also noticed that, the number of common features of documents between two classes is measured for the similarity of these two classes, and the size of positive training data is concerned for avoiding the overfitting problem in classification. Hence, a method that utilizes classification results and the number of feature sets and training sets for constructing a hierarchical taxonomy is proposed. Moreover, three merge-prohibiting conditions are defined for guaranteeing that (1) training sets are consistent with classification models and (2) the overfitting problem is avoided. In our experiments, three real-world data sets, $ModApte$, $20NewsGroups$, and $Edoc$ are used for evaluating classification performance of our method. Also, Support Vector Machines (SVM), proposed by Cortes and Vapnik in 1995 [9], is used as our classification program. Experimental results present the classification accuracy of the predefined hierarchical tree and our method. It is summarized that, although our method does not work well on the dataset in which the distribution of documents in each class is balanced, hierarchical classification accuracy on other datasets is improved by the proposed method, especially when the classes with few documents are ignored.

The rest of the paper is organized as follows: recent research results on hierarchical taxonomy construction are introduced in Section 2. In Section 3, we propose a method to construct a hierarchical taxonomy. Section 4 reports the experimental results on real-world data sets and the performance of the purposed method. Finally, the conclusion is given in Section 5.

2. RELATED WORKS

In this section, hierarchical classification, Support Vector Machine (SVM), and hierarchical taxonomy construction are detailed introduced.

2.1 Hierarchical Classification

A semantically hierarchical taxonomy is similar to the real-world data for classification. For example, in the library, the mathematics and physics belong to the science which is a more general topic. Thus, the mathematics and physics are subclasses of the science,and they are two branches of the science in the hierarchy. For classification, internal nodes of the hierarchy are classifiers and leaf nodes are class labels. An example of hierarchical classification is illustrated in Fig. 1. A document is labeled by using these classifiers at internal nodes and is assigned to their branches until the leaf node is reached. Efficiency is the advantage of hierarchical classification, especially with a large hierarchical taxonomy. When a sample is classified to a branch of the tree, it means that other classifiers, not at this branch, need not be performed. In other words, a large multi-class classification problem could be divided into multiple sub-problems. Indeed, that improves the efficient of classification performance.

There are many different kinds of classifiers employed in the hierarchical tree, like Support Vector Machine [2], [4], naïve Bayes classifier [3], and Association Rules [10]. In [2], the value obtained from the SVM classifier is regarded as the probability of classification. Thus, the probability for each path can be calculated by multiplying the probability at nodes in each path from the root to the leaf node. Then, the path with the highest probability is selected as the classification path, and the sample is labeled as the class at the leaf node in this path.

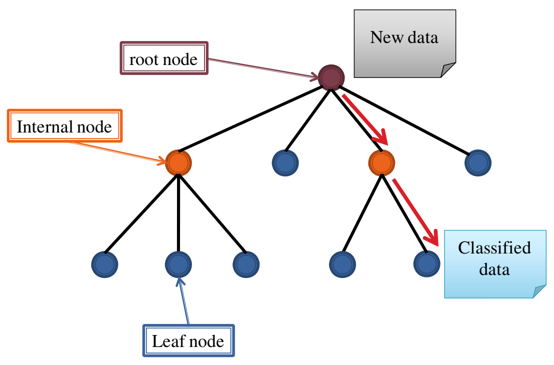

Fig. 1. Hierarchical Classification

2.2 Support Vector Machine (SVM)

The Support Vector Machine is a statistical classification theory purposed by Cortes and Vapnik in 1995 [9]. Its objective function is to find a hyperplane to separate instances with the positive label from ones with the negative label. The optimal hyperplane is decided by maximizing the distance from the nearest training instances to the hyperplane. Basically, SVM is designed for the binary-class classification problems. For the extended multi-class classification problems, One-Against-All (OAA), One-Against-One (OAO), and Error Correcting Output Code (ECOC)[7], have also been discussed in recent years. However, the hierarchical taxonomy construction is our focus in this paper, so the efficiency of SVM multi-class classification is not discussed further. To take OAA as an example, c binary SVM classifiers are trained where c is the number of the classes. SVM classifier i is trained by using all samples in class i as positive samples, and the rest of the samples as negative samples. After these c SVM classifiers are trained, c corresponding decision boundaries are generated. When an unlabeled sample comes, its predicted label is one with the highest decision value.

In SVM training, all training data is set up first. For example, in Fig. 2, there are eight documents in training set where $\{a_1, a_2, a_3\}$ belong to class A, $\{b_1, b_2\}$ belong to class B, and $\{c_1, c_2, c_3\}$ belong to class C. For SVM classifier A, it sets $\{a_1, a_2, a_3\}$ as positive samples and $\{b_1, b_2, c_1, c_2, c_3\}$ as negative samples. For SVM classifier B, it sets $\{b_1, b_2\}$ as positive samples and $\{a_1, a_2, a_3, c_1, c_2, c_3\}$ as negative samples. For SVM classifier C, it sets $\{c_1, c_2, c_3\}$ as positive samples and $\{a_1, a_2, a_3, b_1, b_2\}$ as negative samples. Then SVM classifiers A, B, and C are trained and used for classifying unlabeled documents.

In the flat class hierarchy, each class is a leaf node. Undoubtedly, each node is a SVM classifier for hierarchical models. When a hierarchical taxonomy is

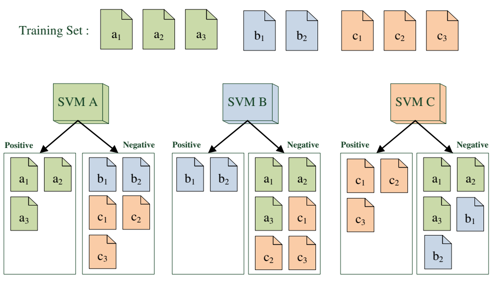

Fig. 2. Train SVM classifiers

provided, the OAA multi-class classification should be performed with some changes. For a SVM classifier at each node, samples in its descendants are set as positive samples, and those in the descendants of its sibling nodes are set as negative ones. In Fig. 3, a hierarchical taxonomy is given based on the example in Fig. 2. For SVM classifier A, its training sets are not changed because it has no descendant and descendants of its sibling node X are class B and C. For SVM classifier X, it is an internal node and it sets $\{b_1, b_2, c_1, c_2, c_3\}$ which are the samples in the descendant classes as positive samples and $\{a_1, a_2\}$ as negative samples. For SVM classifier B, it sets $\{b_1, b_2\}$ as positive samples and $\{c_1, c_2, c_3\}$ as negative samples. For SVM classifier C, it sets $\{c_1, c_2, c_3\}$ as positive samples and $\{b_1, b_2\}$ as negative samples. After all SVM classifiers are trained, an unlabeled sample is classified according to the path which is formed by the nodes with the highest decision value in the classification at each level of the hierarchical taxonomy.

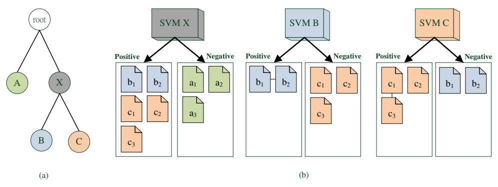

Fig. 3. Train SVM classifiers of hierarchical models. (a) A hierarchical taxonomy. (b) Train SVM classifiers.

2.3 Hierarchical Taxonomy Construction

For convenience, a suitable hierarchical taxonomy should be automatically built by algorithms. Recently, clustering methods are frequently applied for this need. Hierarchical clustering constructs a hierarchy of clusters presented as a tree. Generally, there are two different methods for constructing a hierarchical tree. The first method is the hierarchical agglomerative clustering (HAC) approach, in Fig. 4, which is basically a bottom-up approach from the leaves. It builds the hierarchy by merging individual clusters. Then, the closest elements are merged according to the distance (similarity) between the elements [11], [12]. The second method is the partitioning clustering approach which is a top-down approach from the root and recursively splits the clusters, in Fig. 5. First, all elements are put into the largest cluster. Then, it builds the hierarchy by splitting each cluster into sub-clusters until it reaches the stop criteria or the clusters are singletons. [7] starts from selecting two centroids of the two furthest clusters as initial means and then implements the k-means algorithm [6] with $k = 2$ to partition each cluster into 2 sub-clusters. In [13], authors propose a partition algorithm to split each node into at most n sub-node by minimizing the distance function. $Jensen-Shannon(JS)\ divergence$ [14] is a popular method of measuring the distance (similarity) between two probability distributions, and is used to compute the distance between clusters. However, clustering are usually limited to the predefined parameters, like the number of clusters or the depth of the child nodes. They could not be applied for practical applications.

In [8], authors design an algorithm to adjust the predefined hierarchy. Given a predefined hierarchy, training data, and validation data, the algorithm generates candidate hierarchies by making only one change on the previous hierarchy. After evaluating classification on these hierarchies with validation data, the hierarchy

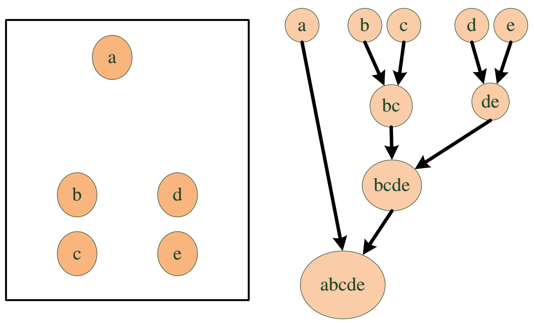

Fig. 4. Hierarchical Agglomerative Clustering

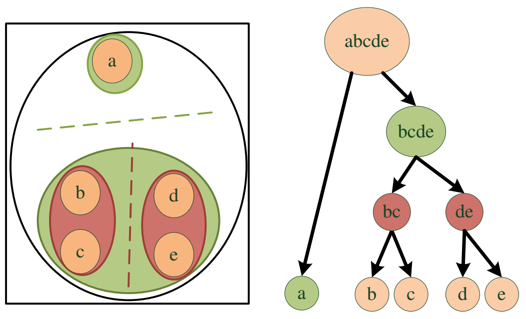

Fig. 5. Hierarchical Partitioning Clustering

with the best improvement is picked. The adjusting hierarchy procedure stops until there is no improvement. However, time is spent on finding the hierarchy according to the performance of validation data. In this paper, in addition to utilizing performance (accuracy/precision) of the validation data, classification results are measured for constructing class hierarchy.

3 Hierarchy Construction based on Learning from Classification Results

In this section, a method of automatically constructing a hierarchical taxonomy based on learning from classification results is proposed. The hierarchical taxonomy is adjusted by merging the classes of which samples are frequently misclassified as each other. We start from the flat model while all leaves are children of the root, and the procedure of our method is described in Fig. 6. Given a hierarchy H, training data T, and validation data V, the procedure is as follow:

1) Train the hierarchical model on T.
2) Evaluate the hierarchical model on V.
3) Adjust the hierarchy H.
4) Repeat Step 1 until no adjustment is made.

The main part of the approach is in Step 3, hierarchy adjustment. The evaluated results are measured to decide which classes should be merged. The measurement in our approach is based on the following observation: biases between classes, feature distribution, and training data size.

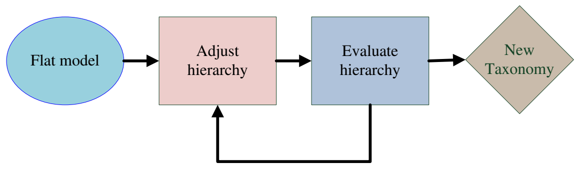

Fig. 6. The Approach for Adjusting Hierarchy

3.1 Biases between Classes

If most of documents in class C_1 are misclassified to class C_2 and most of documents in class C_2 are misclassified to class C_1, class C_1 and C_2 should be merged in order to distinguish documents in them in the new branch more clearly. Thus, the biases between classes are defined as

$$Bias(C_i \rightarrow C_j) = \frac{n}{Ori\_Mis(C_i)} \times \frac{n}{Cla\_Mis(C_j)}$$

where n is the number of documents in class C_i misclassified to class C_j, $Ori\_Mis(C_i)$ is the number of misclassified documents in class C_i, and $Cla\_Mis(C_j)$ is the number of documents which are misclassified to class C_j. The first fraction is the ratio of documents misclassified to C_j in misclassified documents that actually belong to class C_i. The second fraction is the ratio of documents misclassified to C_j in documents misclassified to class C_j.

An example is illustrated in Tab. I where class labels in the column are true label and in the row are predicted labels. The value in the table is the number of documents that belong to a class (column) misclassified to another class (row). There are three class A, B, and C. The bias from class A to B is:

TABLE I

AN EXAMPLE OF BIAS BETWEEN CLASSES.

	A	B	C	Ori_Mis
A	50	40	5	45
B	20	100	10	30
C	10	10	80	20
Cla_Mis	30	50	15	

$Bias(C_A \rightarrow C_B) = \frac{40}{45} \times \frac{40}{50} = 0.711$. On the other hand, the bias from class B to A is: $Bias(C_B \rightarrow C_A) = \frac{20}{30} \times \frac{20}{30} = 0.444$. As we know, the biases between class A and B is not symmetric. For the symmetric biases between classes, the $Score$ value calculated by the biases between class C_i and C_j is defined as:

$$Score(C_i, C_j) = Bias(C_i \rightarrow C_j) \times Bias(C_j \rightarrow C_i)$$

3.2 FEATURE DISTRIBUTION

When there are many common features in two classes, it is difficult to distinguish documents between these two class labels. Thus, these two classes should be merged, such that the difference between these classes can be distinguished more clearly in this new branch. The ratio of common features is defined as:

$$FRatio(C_i, C_j) = \frac{\sum_i f_i \times w_i \times comm(f_i, C_i, C_j)}{\sum_i f_i \times w_i}$$

, where w_i is the weight of feature f_i and $comm(f_i, C_i, C_j)$ is 1 if f_i appears in both class C_i and C_j, otherwise, 0.

3.3 TRAINING DATA SIZE

When there are two classes and 90% training sets belong one of them, classification may be over-fitted with this dominant class. That is because features are significantly weighted if they belong to the dominant class. Thus, in order to avoid the problem of overfitting, the ratio of sizes of positive training data is defined as:

$$TSRatio(C_i, C_j) = \frac{\min PTrainingSize(C_i, C_j)}{\max PTrainingSize(C_i, C_j)}$$

3.4 MERGING CLASSES WITH TENDENCIES (MCT)

According to the above factors, a score function is defined for measuring the needs of merging class nodes:

$$A\_Score(C_i, C_j) = (1 + Score(C_i, C_j))$$

$$\times FRatio(C_i, C_j) \times TSRatio(C_i, C_j)^2$$

, $TSRatio(C_i, C_j)^2$ means that we want to merge the classes with almost equal training data size.

Moreover, it is worth noticing that any two classes should not be merged if one of the following three merge-prohibiting conditions occurred in the hierarchical tree: (A) $Two\text{-}Children$: these two class nodes chosen to be merged have no sibling node. It is not necessary to perform the merging on them. (B) $Different\text{-}Training\text{-}Set$: the training samples in these two classes are changed when previous merging operations are performed. When the training samples are changed, they are not consistent with the classifiers and classification results. Hence, they can not be merged in the current hierarchy. For example, in Fig. 7, class C and D are chosen to be merged. When moving the position of class D to be the sibling of class C, the positive and negative training samples in class X and B are changed. So the class X, B, C, and D can not be merged in adjusted hierarchy in which Class C and D are already merged. (C) $Oversize$: that is used to prohibit the internal node from having too many positive training samples. If it occurs, the problem of overfitting might be caused. The rule is regarded as a threshold to prohibit the growth of subtrees. If the number of positive samples of the internal node is larger than the threshold after its descendant class nodes are merged, the merging is not performed.

In Fig. 8, the number above the nodes is the number of positive samples in each node class. Assume that class C and D are chosen to be merged. If they are merged by moving the position of class D to be the

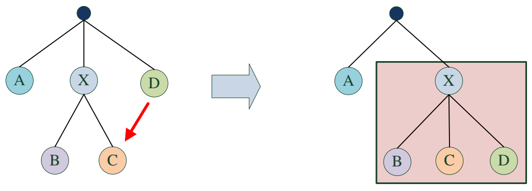

Fig. 7. Example for Rule B.

sibling of class C, the positive samples of class X which is the ancestor of class C will increase. If X has too many positive training samples, there will be a high probability to classify the document to class X, not A. In order to avoid this overfitting problem, Rule C: $Oversize$, that is set as the number of positive samples in class A, is used to prohibit the internal class nodes from having too many positive samples.

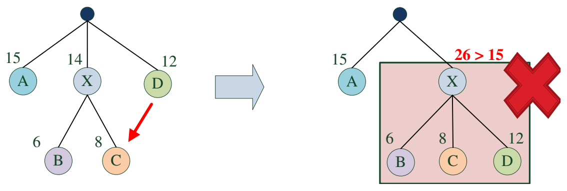

Fig. 8. Example for Rule C.

Therefore, the proposed Merging Classes with Tendencies (MCT), the adjusting procedure for the hierarchy on each round, is introduced step by step in the following.

$Step$ 1. Classification results are evaluated on the current hierarchical taxonomy.

$Step$ 2. Assume that $A\_Score(C_m, C_x)$ is the highest A_Score value of class C_m, and if the $A\_Score(C_m, C_x)$ is also the highest A_Score value of class C_x, it means that documents in class C_m and C_x are frequently misclassified to each other. Thus, C_m and C_x are chosen to be merged.

$Step$ 3. Two cases are considered that class C_m and C_x are at the same level in the hierarchy or not. Class C_m and C_x which are at the same level can be merged immediately. Or, if the class node with higher accuracy is at level 1 in the hierarchy, class C_m and C_x are merged immediately. Otherwise, the position of the class node with lower accuracy is changed to be the sibling node of the other one.

$Step$ 4. Three merge-prohibiting conditions are used to check that the classes could be merged or not. If one of three merge-prohibiting conditions is broken, merging operation on these two class nodes is cancelled.

$Step$ 5. Go back to the $Step$ 2 to find out the classes with the second highest A_Score value. If there is no class to be merged at this round, going to $Step$ 1 to evaluate classification results on the new hierarchy. Our adjusting procedure stops until there is no need to merge class nodes.

4. Experiments

4.1 Performance Testing

TABLE II
THE CONTINGENCY TABLE FOR CLASS C_i

Class C_i	Actual Pos.	Actual Neg.
Predict Pos.	TP_i	FP_i
Predict Neg.	FN_i	TN_i

The standard precision, recall, and F1 measure are used for evaluating the performance of our proposed method. Given the contingency table for class C_i (Tab. II), the precision(P_i), recall(R_i), and F1 measure($F1_i$) of class C_i are evaluated as:

$$P_i = \frac{TP_i}{TP_i + FP_i}, \quad R_i = \frac{TP_i}{TP_i + FN_i},$$

$$F1_i = \frac{2 \times P_i \times R_i}{(P_i + R_i)}$$

There are two ways to aggregate classification accuracy over all classes. One is to average the precision, recall, and F1 measure of each class, called $macroaveraging$. The other is calculated from the global contingency table (Tab. III), called $microaveraging$. $macroaveraing$ is significantly affected by the performance of classes with few documents, and $microaveraging$ is significantly affected by the performance of dominant classes.

TABLE III

THE GLOBAL CONTINGENCY TABLE

Class set $C = C_1, C_2, ..., C_n$	Actual Pos.	Actual Neg.
Predict Pos.	$\sum_{i=1}^{n} TP_i$	$\sum_{i=1}^{n} FP_i$
Predict Neg.	$\sum_{i=1}^{n} FN_i$	$\sum_{i=1}^{n} TN_i$

4.2 DATA SET

Three kinds of data sets, $ModApte$, $20NewsGroups$ [16], and $Edoc$ are used in our experiments, and each document belongs to only one class. The first data set, $ModApte$, is the split of Reuters-21578 collection [15], in which there are 90 classes and 13,331 documents. The second data set, $20NewsGroups$, has 20 classes and 9,414 documents. The third data set, $Edoc$, is collected by Chinese official documents in National Chung Cheng University. $Edoc$ has 81 classes and 5,342 documents. In the environment of our experiments, each data set is partitioned into 70 percent for training and 30 percent for validating. The training data sizes in each class can be regarded as a normal distribution. The standard deviations of training data sizes of each data set are shown in Tab. IV.

TABLE IV

THE DETAILS OF DATA SETS

Data Set	Classes	Train	Validate	Deviation
$ModApte$	90	9586	3745	351.97
$ModApteTop10$	10	7194	2788	842.07
$20NewsGroups$	20	6589	2825	36.36
$Edoc$	81	3738	1604	114.14

4.3 EXPERIMENTAL RESULTS

The term weighting method used for weighting features is TF-IDF (Term Frequency-Inverse Document Frequency) [17], and SVM^{light}[18] is our program of classification. Our method starts from the flat hierarchy and adjusts it based on learning from the classification results at each round, named MCT in the following charts. Also, we terminate the classification paths on the first-level class nodes, named $MCT-Lv1$, in order to emphasize on the classification performance in adjusted hierarchical taxonomies and ignore the problem of ambiguous class boundary. The predefined hierarchical taxonomy which is defined artificially by the experts is given to be compared with our methods, named $Pred$ in charts of experimental results.

Figure 9 and 10 are the macroaveraging and microaveraging accuracy on $ModApte$. It is reported that the accuracy of $MCT-Lv1$ increases since the classes in which documents are frequently misclassified to each other are merged. It is also presented that are only 1% and 1.6% improvement made by our method on macroaveraging and microaveraging accuracy at round 3. In this case, only top 10 dominant classes in $ModApte$ are used, $ModApteTop10$ (9,982 documents). Figure 11 and 12 show that classification accuracy is improved 9% and 1.8% by our method. Classes with few documents are ignored, so the macroaveraging accuracy is significantly improved.

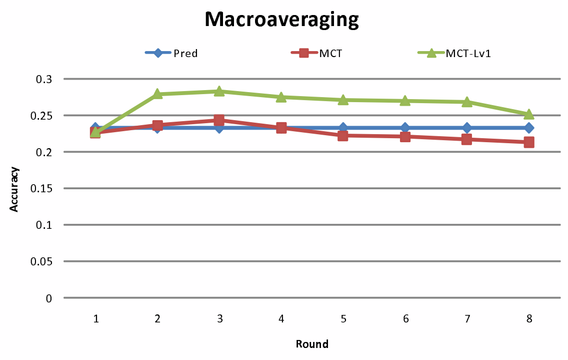

Fig. 9. Macroaveraging accuracy on $ModApte$

According to the numerical analysis on datasets in Tab. IV, the size of training data in each class on $20NewsGroups$ is more balanced than other data sets. There will be no adjustment in the hierarchy because of Rule C that constraints the size of positive training sets. Thus, Rule C is ignored for testing its rationality and allowing our method to adjust the hierarchy on $20NewsGroups$. Figure 13 and 14 are the macroaveraging and microaveraging accuracy on

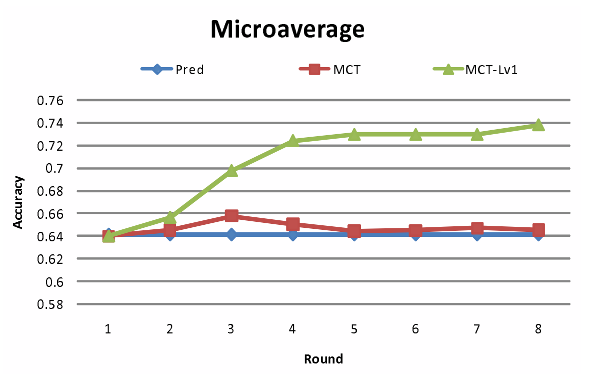

Fig. 10. Microaveraging accuracy on $ModApte$

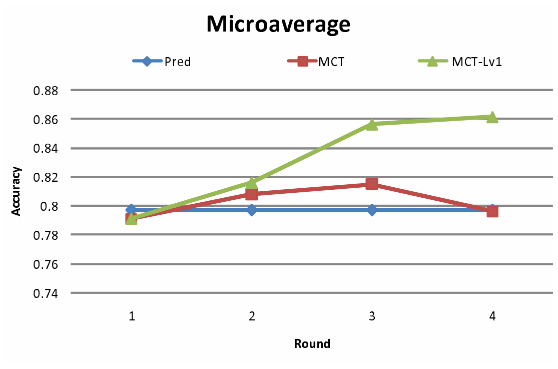

Fig. 12. Microaveraging accuracy on $ModApteTop10$

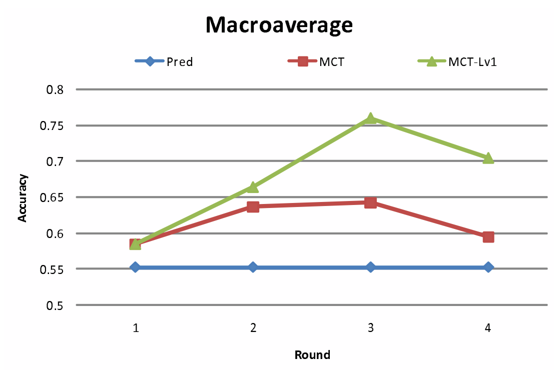

Fig. 11. Macroaveraging accuracy on $ModApteTop10$

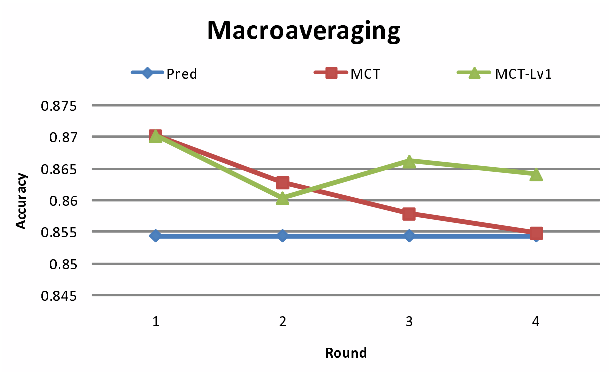

Fig. 13. Macroaveraging accuracy on $20NewsGroups$

$20NewsGroups$. It is shown that classification accuracy of our method decreases from the beginning. The reason is that the overfitting problem decreases the accuracy performed on the adjusted hierarchical taxonomies. Therefore, Rule C is reasonably defined and our method doesn't work well on the balanced datasets.

Figure 15 and 16 are the macroaveraging and microaveraging accuracy on $Edoc$. It is reported that our method improves on macroaveraging accuracy before Round 7. During Round 7 to 10, the macroaveraging accuracy of our method decreases while the microaveraging accuracy of $MCT-Lv1$ increases. That is because two similar sizes of training data of class nodes are merged, and Rule C is almost broken. Therefore, the macroaveraging accuracy decreases because the problem of overfitting occurs. Since macroaveraging accuracy is significantly affected by the performance of classes with few documents, we choose the top 50 and 25 dominant classes in $Edoc$, $EdocTop50$ and $EdocTop25$, for experiments. Figure 17, 18, 19, and 20 show that the decreasing in macroaveraging accuracy is becoming smaller. However, the experimental results show that there are only little improvement made by our method on $Edoc$. $Edoc$ is a collection of Chinese official documents which are composed of Chinese characters and short contents. On $Edoc$, the number of features is 8,141 and each document contains 26.55 features in average. Compared to $Edoc$, on $ModApte$, the number of features is 55,784 and each document contains 53.28 features in average. Because short document contents usually contain insufficient information, the classification on $Edoc$ can not be performed precisely with only few distinct features. Also, there are few common features in the merged class nodes. Hence, it is difficult to classify documents

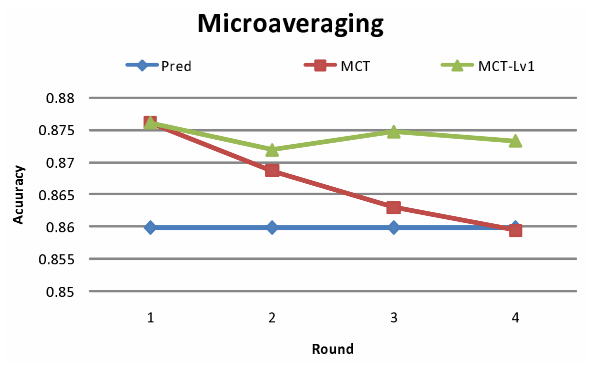

Fig. 14. Microaveraging accuracy on $20NewsGroups$

on *Edoc* correctly at the leaf class nodes because of insufficient features, although the document is correctly classified at the first level node by our merged method.

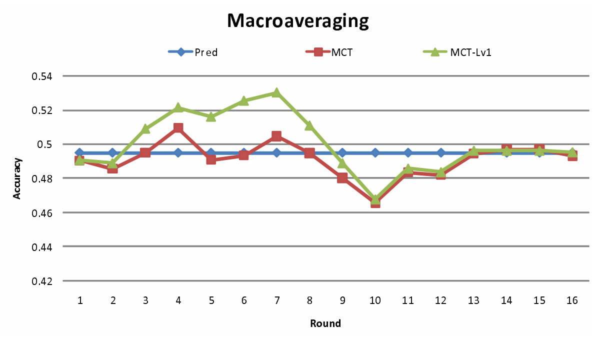

Fig. 15. Macroaveraging accuracy on $Edoc$

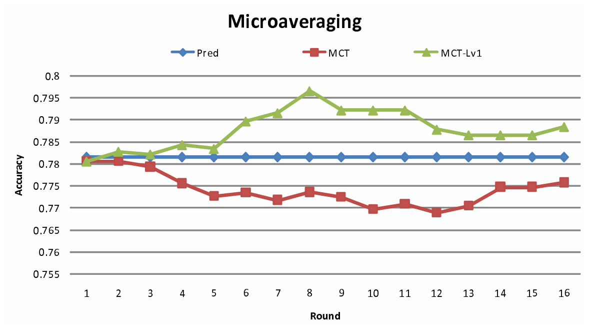

Fig. 16. Microaveraging accuracy on $Edoc$

5. Conclusions

When using hierarchical taxonomies, a large multi-class classification problem are divided into multiple sub-problems. Indeed, the accuracy and efficiency of the classification are significantly improved. In this

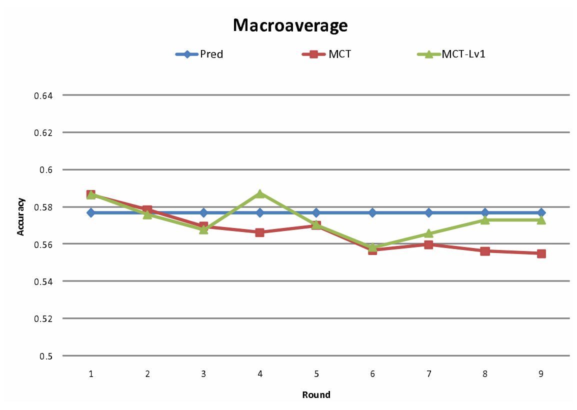

Fig. 17. Macroaveraging accuracy on $EdocTop50$

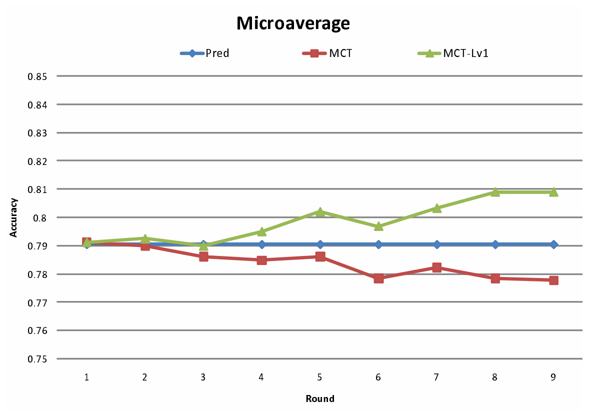

Fig. 18. Microaveraging accuracy on $EdocTop50$

paper, we propose a method based on learning from classification results to adjust the predefined hierarchical taxonomy. The classification results of validation data are used to merge the classes in which documents are frequently misclassified to each other. That could enhance the classification performance in new branches of the class hierarchy. Detailedly, three factors, biases between classes, feature distribution, and training data size, are applied for measuring the need of merging classes. Also, three merge-prohibiting conditions are designed for guaranteeing that (1) training sets are consistent with classification models and (2) the overfitting problem is avoided. In the experiments, our results show that macroaveraging/microaveraing accuracy is improved 1%/1.6% and 9%/1.8% by our method on $ModApte$ and $ModApteTop10$, respectively. However, our method doesn't work well on $20NewsGroup$

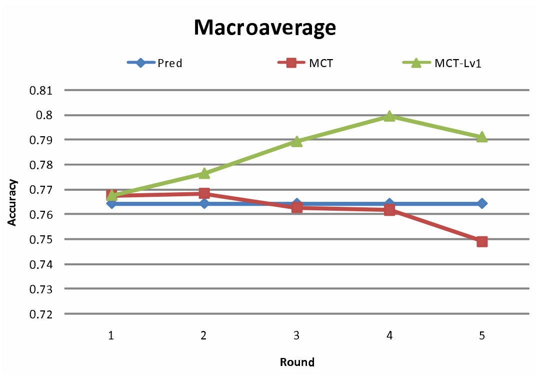

Fig. 19. Macroaveraging accuracy on $EdocTop25$

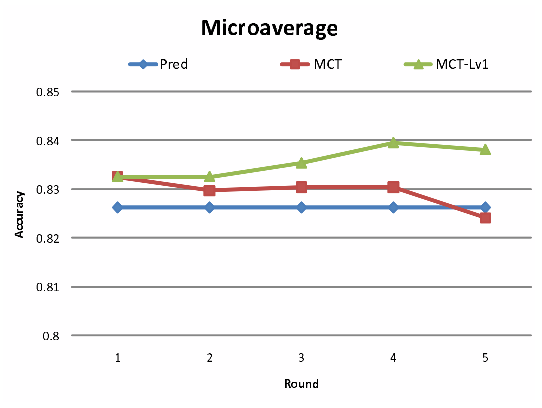

Fig. 20. Microaveraging accuracy on $EdocTop25$

because the distribution of documents in each class is balanced. That is not a common property in real-world datasets. And, there is only little improvement on $Edoc$ since there are not sufficient feature sets. In summary, our method is practically useful to adjust the predefined hierarchy based on learning from classification results. In the future, formula of the factors for measuring needs to merge classes should be studied more detailedly. That could improve the accuracy and efficiency of classification on the hierarchical taxonomy. Also, a suitable stop criteria should be carefully defined in order to gain the highest classification accuracy when the adjusting procedure stops.

REFERENCES

[1] Wikipedia. Available: http://www.wikipedia.org/

[2] S. Dumais and H. Chen. "Hierarchical classification of web content," *in Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 256-263, 2000.

[3] D. Koller and M. Sahami. "Hierarchically classifying documents using very few words," *in Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 170-178, 1997.

[4] T. Y. Liu, Y. M. Yang, H. Wan, H. J. Zeng, Z. Chen, and W. Y. Ma. "Support vector machines classification with a very large-scale taxonomy", *ACM SIGKDD Explorations Newsletter*, vol. 7(1), pp. 36-43, 2005.

[5] L. Cai and T. Hofmann. "Hierarchical document categorization with support vector machines," in Proceedings of the thirteenth ACM international conference on Information and knowledge management, pages 78V87, 2004.

[6] J. A. Hartigan. *Clustering Algorithms*, Wiley, 1975.

[7] K. Punera, S. Rajan, and J. Ghosh. "Automatically learning document taxonomies for hierarchical classification," *Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 1010-1011, 2005.

[8] L. Tang, J. P. Zhang, and H. Liu. "Acclimatizing Taxonomic Semantics for Hierarchical Content Classification," *in Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 384-393, 2006.

[9] C. Cortes and V. Vapnik. *The Nature of Statistical Learning Theory*, Springer, 1995.

[10] K. Wang, S. Q. Zhou, and S. C. Liew. "Building hierarchical classifiers using class proximity", *in Proceedings of the 25th International Conference on Very Large Data Bases Conference*, pages 363-374, 1999.

[11] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*, Prentice Hall, Englewood Cliffs, NJ, 1988.

[12] M. Aghagolzadeh , H. Soltanian-Zadeh, B. Araabi, A. Aghagolzadeh. "A Hierarchical Clustering Based on Mutual Information Maximization", *IEEE International Conference on Image Processing*, vol. 1, pp. 277-280, 2007.

[13] K. Punera, S. Rajan, and J. Ghosh. "Automatic Construction of N-ary Tree Based Taxonomies," *in Proceedings of the Sixth IEEE International Conference on Data Mining*, pp. 75-79, 2006.

[14] I. S. Dhillon, S. Mallela, and R. Kumar. "Enhanced word clustering for hierarchical text classification", *in Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 191-200, 2002.

[15] D. Lewis. Reuters-21578 Text Categorization Test Collection, Distribution 1.0, Manuscript, 1997. Available: http://www.daviddlewis.com/resources/testcollections/reuters21578.

[16] T. Mitchell. 20 Newsgroups. Available: http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html.

[17] G. Salton, C. Buckley. "Term Weighting Approaches in Automatic Text Retrieval", *Information Processing and Management*, vol. 24(5), pp. 513-523, 1988.

[18] T. Joachims. SVM^{light}. Available: http://svmlight.joachims.org/