Hierarchical Classification of Real Life Documents

***Ke Wang*[*], *Senqiang Zhou*[†], *Yu He*[‡]**

1 Introduction

Two features have successfully made on-line information comprehensible and accessible to people: hierarchically structured classes where topics are organized into a hierarchy of increasing specificity, and multi-classed documents where a document is classified into all relevant classes. One such information source is Yahoo! where a document on *Dance*, for example, could be reached from both *Arts* : *Performing_Arts* and *Recreation* topics in the topic hierarchy. The hierarchical feature of classes allows information to be examined and browsed at various topic specificities, and the multi-class feature allows information to be accessed from all related topics. However, most document classification techniques assume that there is a flat class space and each document has one class. The documents classified by such techniques are difficult to browse and access by people, especially when there are many classes such as in Yahoo!. In this paper, we propose a new technique for automatic classification of documents to address these real life requirements. This raises several research issues. We use Yahoo! for explanation.

1. *Misclassification is non-symmetric.* Misclassifying an ads on topic $Travel$ into topic $Outdoors$ is less erroneous than misclassifying it into topic $Software$. Indeed, the fact that many ads belong to both $Travel$ and $Outdoors$, but few belong to both $Travel$ and $Software$ suggests that $Travel$ is more similar to $Outdoor$ than to $Software$. This feature becomes more prevailing in a hierarchical class space where some classes are more general than others.

[*]Simon Fraser University
[†]Simon Fraser University
[‡]National University of Singapore

2. *Documents are multi-classed.* A document is typically classified into all relevant classes. Traditional classification fails to do so because each training document is allowed to be associated with only one class. One solution to multi-classification is to build a classifier for each possible class and classify a new document by going through every classifier. In a sparsed class space where a document belongs to a small number of classes from a large class space, this approach will construct too many classifiers. Further, such an independent classification for each class does not take into account the similarity and hierarchical structure of classes as discussed above.

3. *The sparse class space.* For k classes, a document could be associated potentially with any one of the $2^k - 1$ subsets of classes. Even not so large k will create a very sparse class space in multi-class classification. This sparseness makes the learning task of automatic classification difficult because there may not be enough training documents. However, unlike traditional classification, classes share similarities as discussed above. Exploring such similarities opens up new channels to deal with the sparsity problem of the class space.

In summary, classes of documents interact by being a generalization of one another and classifying common documents. Traditional classification techniques fail to recognize such interactions. In this paper, we regard classes as objects whose similarity can be measured, and the goal of classification is to determine the set of "relevant" classes under this measure. We consider training documents of the form $\{t_1, \ldots, t_n | C_1, \ldots, C_k\}$, where $t_1, \ldots, t_n$ are terms (keywords or phrases) from a given universe and $C_1, \ldots, C_k$ are classes from a given class hierarchy. $\{C_1, \ldots, C_k\}$ is called a *classset*. Given a collection of training documents, our task is to construct a classifier, consisting of rules of the form $\{t_{i_1}, \ldots, t_{i_p}\} \rightarrow \{C_{i_1}, \ldots, C_{i_q}\}$, that assigns a "good" classset $\{C_{i_1}, \ldots, C_{i_q}\}$ to a given new document. There are two contributions:

1. We define a new notion of similarity between two classsets using the similarity of the documents belonging to these classsets. The intuition is that two classsets are similar exactly when their classified documents are similar. Indeed, if two classsets classify many documents in common, the chance that they are similar topics is high, and misclassification from one to the other is less erroneous. We believe that this notion captures the essence of class similarity.

2. We construct a classifier using the proposed class similarity. A major challenge is the search of classification rules of the form $\{t_{i_1}, \ldots, t_{i_p}\} \rightarrow \{C_{i_1}, \ldots, C_{i_q}\}$, where t_{i_j} are terms and C_{i_j} are classes, because there are many terms and classes. Our approach is to apply the association rule mining [1, 2] to generate such rules. We present an algorithm for selecting association rules to construct a classifier.

We evaluate this method using the documents in ACM Digital Library and Yahoo!.

2 Related work

With few exceptions, most classification systems assume that all classes are at a flat level and each document is labeled by one class [5, 6, 9]. Recently, hierarchically structured classes were examined in [3, 4, 7] where classes are organized into a hierarchy of increasing specificity and a document is labeled by one class in the hierarchy. Though a document belonging to a child class is automatically considered as belonging to a parent class, a document is not allowed to belong to two classes not on a generalization path in the hierarchy, like $Arts : Performing\_Arts$ and $Recreation$ in Yahoo!.

Related to multi-classification of documents is the problem of transforming source terms to target terms for a collection of documents [8]. Each training document is a pair $< s, t >$ where s is a set of source terms and t is a set of target terms. The goal of the transformation is finding the transformation matrix from source terms to target terms that minimizes the total error for a collection of documents. In our terminology, target terms correspond to classes of a document and the transformation matrix corresponds to a classifier. [8] solves this problem as the Linear Least Squares Fit that performs the transformation using the standard singular value decomposition. That approach does not address the similarity and hierarchical nature of target terms. Also, the singular value decomposition is computationally expensive, in order $N^2 \times k^3$, where N is the number of terms and documents and k is the number of terms. For ACM Digital Library, N and k could be tens of thousands or even more.

The extended kNN [9] returns the set of classes of the k training documents nearest to the given document as the relevant classes. We will compare our method with kNN. Some classifiers such as decision tree [5] return a class distribution for a given document, in the form of the probability of each class. However, these classifiers do not consider the similarity of classes and the hierarchical structure of the class space.

3 A new class similarity

We measure the similarity of two classsets by the similarity of the training documents belonging to them. Let us describe this idea formally. Consider a document d belonging to k classes $C_1, \ldots, C_k$, or simply belonging to the classset $CS = \{C_1, \ldots, C_k\}$. Clearly, d also belongs to the classes that are more "general" than C_i, $1 \leq i \leq k$. These general classes are the ancestors of C_i in the class hierarchy. Let $Anc(CS)$ denote the set of classes in CS plus their ancestors.

Definition 1. *Consider two classsets CS_1 and CS_2. We say that CS_1 is more general than CS_2 if $Anc(CS_1) \subseteq Anc(CS_2)$. We say that a document d is covered by a classset CS if CS is more general than the classset of d. The coverage of CS, denoted by $Cover(CS)$, is the set of all documents covered by CS.*

For example, if $Dance$ is a parent of $Fast\_Dance$, $\{Dance\}$ is more general than $\{Fast\_Dance, Music\}$ because $Anc(\{Dance\}) \subseteq Anc(\{Fast\_Dance, Music\})$.

The following equivalence holds (we omit the straightforward proof):

Lemma 2. *Consider a classset CS and a document d with classset CS_d. The following are equivalent: (1) $d \in Cover(CS)$; (2) $CS \subseteq Anc(CS_d)$; (3) $Anc(CS) \subseteq Anc(CS_d)$.*

Lemma 3. $Cover(CS_1) \cap Cover(CS_2) = Cover(CS_1 \cup CS_2)$.

Proof: Let $d \in Cover(CS_1) \cap Cover(CS_2)$. From Lemma 2, $CS_1 \subseteq Anc(CS_d)$ and $CS_2 \subseteq Anc(CS_d)$, where CS_d denotes the classset of d, therefore, $CS_1 \cup CS_2 \subseteq Anc(CS_d)$. From Lemma 2, this implies that $d \in Cover(CS_1 \cup CS_2)$. This shows that $Cover(CS_1) \cap Cover(CS_2) \subseteq Cover(CS_1 \cup CS_2)$. To show the other containment, let $d \in Cover(CS_1 \cup CS_2)$. From Lemma 2, $CS_1 \cup CS_2 \subseteq Anc(CS_d)$, and thus, $CS_i \subseteq Anc(CS_d)$, $i = 1, 2$. Then, from Lemma 2, d is in $Cover(CS_i)$.

The *dissimilarity* of two classsets CS_1 and CS_2 is defined as the normalized difference of their coverages:

$$E(CS_1, CS_2) = \frac{|Cover(CS_2) - Cover(CS_1)| + |Cover(CS_1) - Cover(CS_2)|}{|Cover(CS_1) \cup Cover(CS_2)|} \quad (1)$$

$|x|$ denotes the number of elements in a set x. $E(CS_1, CS_2)$ is in the range [0,1]. The *similarity* of CS_1 and CS_2 is defined as $1 - E(CS_1, CS_2)$. From Lemma 3, we rewrite definition (1) as

$$E(CS_1, CS_2) = \frac{|Cover(CS_1)| + |Cover(CS_2)| - 2|Cover(CS_1 \cup CS_2)|}{|Cover(CS_1)| + |Cover(CS_2)| - |Cover(CS_1 \cup CS_2)|} \quad (2)$$

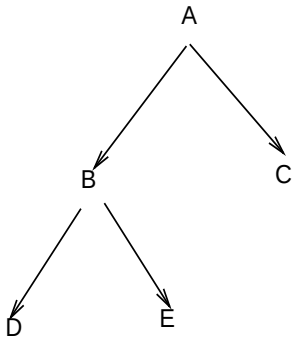
Therefore, to compute $E(CS_1, CS_2)$, we need only to compute the coverage of $CS_1, CS_2, CS_1 \cup CS_2$.

We say that a document d *matches* a rule $T \rightarrow CS$, or vice versa, if d contains all the terms in T. Let $Match(T \rightarrow CS)$ denote the set of training documents that match $T \rightarrow CS$. The *generalized confidence* of $T \rightarrow CS$ is defined as

$$Conf_g(T \rightarrow CS) = \frac{Match(T \rightarrow CS) - \Sigma_d E(CS_d, CS)}{Match(T \rightarrow CS)} \quad (3)$$

where d ranges over the elements of $Match(T \rightarrow CS)$, and CS_d is the classset of d. Intuitively, $Conf_g(T \rightarrow CS)$ measures the average similarity between CS and the classsets of the documents that match $T \rightarrow CS$. If $E(CS_1, CS_2)$ is binary, i.e., 1 or 0, $\Sigma_d E(CS_d, CS)$ degenerates to the number of training documents that match $T \rightarrow CS$ but do not belong to CS, and $Conf_g(T \rightarrow CS)$ degenerates to the standard confidence [1]. In this sense, $Conf_g(T \rightarrow CS)$ is a generalization of the standard confidence in the presence of class similarity.

Example 1. *Consider the database of 6 documents $d_1, \ldots, d_6$ and the class hierarchy in Figure 1. We write BC to mean classset $\{B, C\}$, and similarly for the others. $Cover(B) = \{d_1, d_2, d_3, d_4, d_6\}$ because B is contained in $Anc(CS_{d_i})$ for $i = 1, 2, 3, 4, 6$ (from Lemma 2). Figure 1(c) lists the coverage of some classsets. For example, $E(B, C)$ is computed as follows:*



(a) Class hierarchy

id	items	classset
d_1	k_1	B
d_2	k_1, k_2	BC
d_3	k_1, k_2	BC
d_4	k_2	CD
d_5	k_3	C
d_6	k_3	CE

(b) Documents

CS	$Cover(CS)$
B	d_1, d_2, d_3, d_4, d_6
BC	d_2, d_3, d_4, d_6
C	d_2, d_3, d_4, d_5, d_6
CD	d_4
CE	d_6

(c) Coverage

Figure 1. *An example*

$$Cover(B) - Cover(C) = \{d_1, d_2, d_3, d_4, d_6\} - \{d_2, d_3, d_4, d_5, d_6\} = \{d_1\}$$
$$Cover(C) - Cover(B) = \{d_2, d_3, d_4, d_5, d_6\} - \{d_1, d_2, d_3, d_4, d_6\} = \{d_5\}$$
$$Cover(B) \cup Cover(C) = \{d_1, d_2, d_3, d_4, d_5, d_6\}$$

$$E(B, C) = \frac{|\{d_1\}| + |\{d_5\}|}{|\{d_1, d_2, d_3, d_4, d_5, d_6\}|} = \frac{2}{6} = 0.33$$

Table 1 shows the dissimilarity between some classsets.

$Conf_g(k_1 \rightarrow B)$ is computed as follows. k_1 is found in d_1, d_2, d_3, 1 with no error and 2 with error $E(B, BC)$. So we have

$$Conf_g(k_1 \rightarrow B) = \frac{3 - 2 \times E(B, BC)}{3} = \frac{3 - 2 \times 0.2}{3} = 0.87$$

Similarly,

$$Conf_g(k_2 \rightarrow BC) = \frac{3 - E(BC, CD)}{3} = \frac{3 - 0.75}{3} = 0.75$$

$$Conf_g(k_3 \rightarrow C) = \frac{2 - E(CE, C)}{2} = \frac{2 - 0.8}{2} = 0.6$$

$$Conf_g(k_2 \rightarrow C) = \frac{3 - 2 \times E(BC, C) - E(CD, C)}{3} = \frac{3 - 2 \times 0.2 - 0.8}{3} = 0.6$$

	B	BC	C	CD	CE
B	0	0.2	0.33	0.8	0.8
BC	0.2	0	0.2	0.75	0.75
C	0.33	0.2	0	0.8	0.8
CD	0.8	0.75	0.8	0	1
CE	0.8	0.75	0.8	1	0

Table 1. $E(CS_1, CS_2)$

4 Construction of classifiers

There are four steps in constructing a classifier. First, we generate association rules of the form $T \rightarrow CS$, where T is a set of terms and CS is a classset, that satisfy the user-specified minimum support and minimum confidence, as in [1, 2]. Second, we rank rules to determine the classification rule of a document. Third, we remove the rules that incorrectly classify many training documents. Fourth, we cut off the ranked list to minimize the overall classification error. Let us explain each step in details.

4.1 Step 1: Find association rules

We generate all association rules of the form $T \rightarrow CS$ that satisfy some user-specified minimum support and minimum confidence. The algorithm is basically that of [2]. There are several differences.

First, each frequent k-itemset ([2]'s terminology), $k > 1$, contains at least one term and at least one class. Each such frequent itemset TCS represents a rule $T \rightarrow CS$, where T is a set of terms and CS is a classset. For $k > 2$, every frequent k-itemset of this form can be constructed using two frequent $(k-1)$-itemsets of the same form, like in the Apriori [2], adding either one term in T or one class in CS. This restricted form reduces substantially the number of itemsets generated. Second, we use the generalized confidence of rule $T \rightarrow CS$ as in defined by Equation 3, not the standard confidence in [2]. $E(CS, CS_d)$ in Equation 3 is not available from frequent itemsets. We need two database scans to compute generalized confidence of all rules.

1. In the first scan, we compute $|Cover(CS)|$ for CS, where CS is either a classset CS_d appearing in some training documents, or a classset CS_r appearing in some rules found, or $CS_d \cup CS_r$. In particular, for each document d scanned, we increment $|Cover(CS)|$ for CS if $CS \subseteq Anc(CS_d)$, where CS_d is the classset of d. Then, we compute $E(CS_d, CS_r)$ based on Equation (1) for classset CS_d of training documents d and classset CS_r in rules r found.

2. In the second scan, we compute $Conf_g(T \rightarrow CS)$ for every rule $T \rightarrow CS$ generated. In particular, for each training document d, we find all matching rules $T \rightarrow CS$, i.e., $T \subseteq d$, increment $Match(T \rightarrow CS)$ and add $E(CS_d, CS)$ to $\Sigma_d E(CS_d, CS)$. At the end of the scan, $Conf_g(T \rightarrow CS)$ is computed by Equation (3).

4.2 Step 2: rank the rules

A document could match more than one rule, one of which is chosen to classify the document. We propose the *most-confident-first (MCF) principle* to determine the classification rule of each document: a document is classified by the matching rule that has the highest generalized confidence (breaking a tie arbitrarily). If we rank all rules by generalized confidence, the MCF principle says that the first matching rule in the ranked list is chosen as the classification rule for a document. The generalized confidence of a rule $T \rightarrow CS$ measures the average similarity between CS and the classsets of the documents matched by the rule. By choosing the matching rule with highest generalized confidence, a document is assigned the most similar classset among all the matching rules. Under the MCF principle, among all the rules $T \rightarrow CS$ with the same LHS T, only the rule with the highest generalized confidence will actually classify some documents, therefore, needs to be kept.

4.3 Step 3: remove rules of low accuracy

Let D be the set of training documents classified by rule $T \rightarrow CS$ under the MCF principle. If D is non-empty, the *accuracy* of $T \rightarrow CS$ is defined as

$$Accu(T \rightarrow CS) = \frac{|D| - Error(T \rightarrow CS)}{|D|} \tag{4}$$

where $Error(T \rightarrow CS)$ is the *error* of $T \rightarrow CS$ defined as

$$Error(T \rightarrow CS) = \Sigma_{d \in D} E(CS_d, CS) \tag{5}$$

The accuracy of all rules can be computed by one scan of the documents, given that all errors $E(CS_d, CS)$ were computed in Step 1. We remove all rules with accuracy below a certain threshold because they contribute negatively to the overall accuracy. Note that $Conf_g(T \rightarrow CS)$ is defined with respect to all documents that match the rule, whereas $Accu(T \rightarrow CS)$ is defined with respect to the documents classified by the rule under the MCF principle.

Example 2. *Consider the database in Example 1 and the error in Table 1. Let the minimum support be 2/6. In Step 1, the following rules above the minimum support are generated (we do not specify minimum confidence), ranked by generalized confidence:*

$r_1 : k_1 \rightarrow B$ *(match* d_1, d_2, d_3*,* $conf_g = 0.87$*)*
$r_2 : k_2 \rightarrow BC$ *(match* d_2, d_3, d_4*,* $conf_g = 0.75$*)*
$r_3 : k_3 \rightarrow C$ *(match* d_5, d_6*,* $conf_g = 0.60$*)*
$r_4 : k_2 \rightarrow C$ *(match* d_2, d_3, d_4*,* $conf_g = 0.60$*)*

let $r_1, \ldots, r_m$ be the remaining rules after Step 3, sorted by $Conf_g(r_i)$
recompute $Error(r_i)$ for $1 \leq i \leq m$
/* compute the cutoff point */
$PrefixError(r_i) = 0$
foreach $i = 1$ to m **do**
$PrefixError(r_i) = PrefixError(r_{i-1}) + Error(r_i)$
let $Unclassifed(r_i)$ be the set of training documents not classified by $r_1, \ldots, r_i$
let $DefaultClass(r_i)$ be the classset CS that minimizes
$\Sigma_{d \in Unclassified(r_i)} E(CS, CS_d)$
$DefaultError(r_i) = \Sigma_{d \in Unclassified(r_i)} E(DefaultClass(r_i), CS_d)$
find the smallest k that minimizes $PrefixError(r_k) + DefaultError(r_k)$
return prefix $r_1, \ldots, r_k$ and default class $DefaultClass(r_k)$

Table 2. *Step 4*

$Conf_g$ is computed as in Example 1. In Step 2, we apply the MCF principle to determine the classification rule for each training document:

$r_1 : k_1 \rightarrow B$ *(classify* d_1, d_2, d_3, $Accu = 0.87$*)*
$r_2 : k_2 \rightarrow BC$ *(classify* d_4, $Accu = 0.25$*)*
$r_3 : k_3 \rightarrow C$ *(classify* d_5, d_6, $Accu = 0.60$*)*
$r_4 : k_2 \rightarrow C$ *(classify no document)*

In Step 3, we compute the accuracy of rules. $Accu(r_1) = Conf_g(r_1) = 0.87$. *$r_2$ classifies only d_4, so* $Accu(r_2) = \frac{1-E(BC,CD)}{1} = 0.25$. *$r_3$ classifies all documents it matches, so* $Accu(r_3) = Conf_g(r_3) = 0.60$. *$r_4$ has no turn to classify any document. Suppose that we set the threshold of accuracy at 0.5, r_2 is removed, and r_4 now classifies d_4.*

4.4 Step 4: cut off the ranked list

Finally, we cut off the ranked list of remaining rules to minimize the cutoff error. Let $r_1, \ldots, r_m$ be the ranked list of remaining rules. Suppose that we cut off the list after the first i rules, $r_1, \ldots, r_i$. The *cutoff error* is $PrefixError(r_i) + DefaultError(r_i)$. $PrefixError(r_i)$ is the sum of the rule error $Error(r_j)$ for all rules r_j, $1 \leq j \leq i$. $DefaultError(r_i)$ is the error caused by assigning the *default classset* to all the training documents not classified by any rule r_j, $1 \leq j \leq i$. The default classset is chosen to minimize $DefaultError(r_i)$. Table 2 shows Step 4. Since the rule error $Error(r_j)$ may have been changed by removing rules in Step 3, In Table 2, we first compute the rule error as in Step 3.

Example 3. *We continue with Example 2. After removing r_2, the error of the remaining rules is computed as follows*

$$Error(r_1) = E(B, CS_{d_1}) + E(B, CS_{d_2}) + E(B, CS_{d_3})$$
$$= E(B, B) + E(B, BC) + E(B, BC) = 0.4$$
$$Error(r_3) = E(C, CS_{d_5}) + E(C, CS_{d_6}) = E(C, C) + E(C, CE) = 0.8$$
$$Error(r_4) = E(C, CS_{d_4}) = E(C, CD) = 0.8$$

In Step 4, we determine the cutoff point of the remaining rules $< r_1, r_3, r_4 >$. *For the shortest prefix* $<>$, *the default classset is* BC *(of* d_2*), which has the minimum error of 1.9. This is shown in the first row of Table 3. For prefix* $< r_1 >$, *the cutoff error is the sum of* $Error(r_1)$ *and the default error for unclassified documents* d_4, d_5, d_6. *The default classset is* C *(of* d_5*), which gives the minimum default error,* $E(C, CD) + E(C, CE) = 1.6$. *Thus, the cutoff error for* $< r_1 >$ *is* $0.4 + 1.6 = 2.0$, *as shown in the second row of Table 3. For prefix* $< r_1, r_3 >$, *the cutoff error is* $Error(r_1) + Error(r_3)$ *plus the default error for unclassified* d_4. *In this case the default classset is the classset of* d_4, CD, *with the default error of 0. So the cutoff error for* $< r_1, r_3 >$ *is 1.2, shown in the third row of Table 3. At this point, since the default error is 0, the cutoff error cannot be reduced by considering longer prefixes. Therefore,* $< r_1, r_3 >$ *is the shortest prefix that has the minimum cutoff error.*

If we do not remove r_2 *in Step 3. The error of each rule in* $< r_1, r_2, r_3, r_4 >$ *is*

$$Error(r_1) = E(B, CS_{d_1}) + E(B, CS_{d_2}) + E(B, CS_{d_3}) = 0.4$$
$$Error(r_2) = E(BC, CS_{d_4}) = E(BC, CD) = 0.75$$
$$Error(r_3) = E(C, CS_{d_5}) + E(C, CS_{d_6}) = E(C, C) + E(C, CE) = 0.8$$
$$Error(r_4) = 0$$

Table 4 shows the computation of cutoff errors. In this case, the empty prefix $<>$ *with the default classset* B *gives the minimum cutoff error, 1.9. This is larger than that of the classifier* $< r_1, r_3 >$ *found earlier.*

5 Experiments

We evaluate the effectiveness of the proposed method using the IBM Patent data and ACM Digital Library. For comparison, most traditional classification methods deal with data in the form of a table or assumes that a document belongs to one class. Such methods cannot work on the multi-classed documents here. We compare our method, denoted Coverage, with two methods, Confidence and kNN. Confidence is the same as Coverage except that it treats each classset as a new class in a flat class space, thus, ignoring the similarity of classset. This method ranks rules by the traditional confidence. Comparison with Confidence will reveal the effectiveness of the proposed similarity of classsets. kNN is the kNN extended with feature selection, which is highly competitive even compared with sophisiticated methods [9]. Given a new document, kNN uses the classes of the k nearest training documents to predict the classset of the new document. The distance of these documents is used as a weight for their classes. One parameter of kNN is *the feature threshold* used by the feature selection. Another parameter of kNN is *the cutoff threshold of class list*. The kNN returns a list of ranked classes (by weight). We select the top classes that are

$< r1, \ldots, ri >$	$Error(ri)$	unclassified doc.	default classset	default error	cutoff error
$<>$	0	$d_1, \ldots, d_6$	BC	1.9	1.9
$< r_1 >$	0.4	d_4, d_5, d_6	C	1.6	2.0
$< r_1, r_3 >$	0.8	d_4	CD	0.0	1.2

Table 3. *The cutoff error for each prefix of* $< r_1, r_3, r_4 >$

$< r1, \ldots, ri >$	$Error(ri)$	unclassified doc.	default classset	default error	cutoff error
$<>$	0	$d_1, \ldots, d_6$	BC	1.9	1.9
$< r_1 >$	0.4	d_4, d_5, d_6	C	1.6	2.0
$< r_1, r_2 >$	0.75	d_5, d_6	C	0.8	1.95
$< r_1, r_2, r_3 >$	0.8			0	1.95

Table 4. *The cutoff error for each prefix of* $< r_1, r_2, r_3, r_4 >$

within the p weight percentile. These are the classes on the top of the list whose total weight is equal to p percentage of the total weight of the whole list. For all methods, the error on a testing document is measured by Equation (1). All results are the average of the 5-fold cross-validation.

5.1 The data sets

The IBM Patent data set (http://www.patents.ibm.com/patlist?xcl=0/). This database contains patent documents categorized by branches and sub-branches. We use branch 451 (Abrading) with 39 sub-branches, and branch 051 (Abrasive tool making process, material, or composition) with 14 sub-branches. For each patent document, we use terms only in Title, Inventor, Abstract and Current class. A class has the form of branch/sub-branch. For example, 451/430 denotes the class corresponding to branch 451 and sub-branch 430. Most documents are associated with one class, and the rest are associated with two or more classes.

The ACM data set (http://www.acm.org/dl/toc.html). This data set maintains a 4-level hierarchical classification of computing related papers. We use level-1 and level-2 topics as the class hierarchy and add level-3 and level-4 topics as terms to documents. Each document is associated with the set of level-1 and level-2 topics of the document. We remove the documents whose classsets appear in less than 15 documents.

Table 5 shows some statistics of the two data sets after the above processing. The partitioning of training documents and testing documents is determined by the 5-fold cross validation.

	ACM	IBM Patent
Documents	15981	4974
Classsets	288	191
classes	40	21
terms	9590	14991
level of class hierarchy	3	3
average size of documents	10.80	45
average size of classsets	1.86	1.55
training documents	12784	3979
testing documents	3197	995

Table 5. *The statistics of the processed data sets*

5.2 The result on IBM Patent data set

Figure 2 shows the classification error over the 995 testing documents. For example, the error of 220 means that the average error of classifying each of the 995 testing documents is 220/995=0.22, which is the difference of the observed classset and the predicted classset, or the difference of the documents belonging to these classsets. On the left side is the error of Coverage and Confidence. The x-axis denotes the minimum support for mining association rules. Different figures correspond to different accuracy thresholds for selecting rules in Step 3. On the right side is the error of kNN. The x-axis denotes the parameter k. Different figures correspond to different feature thresholds. Different curves correspond to different cutoff thresholds of class list. Several observations follow.

The error. Coverage performs significantly better than Confidence, i.e., reducing the error up to 67%. Two factors contribute to this difference. First, Coverage searches for all rules determining a subset of the classset in a training document, but Confidence does not because it treats each classset in the training documents as a new class. As a result, Confidence generates few rules that satisfy the given minimum support, which can be seen from Figure 3, and classification often is done by the default rule. Another reason is that Confidence ignores the similarity of classes, thus, makes no attempt to assign a document to a more similar class in the case of misclassification. The experiment also shows that the error of Coverage is sensitive to the minimum support, but not to the accuracy threshold. Using a small minimum support, Coverage is about 10% to 30% better than the best kNN result.

The size of classifier. On the left side of Figure 3 is the size of the classifiers constructed by Coverage and Confidence. For both methods, the minimum support and the accuracy threshold affects the size. The experiment suggests that minimum support of 1% and accuracy threshold of 2% give a classifier that is both accurate and small.

The execution time. As shown on the right side of Figure 3, Coverage takes longer time than Confidence due to computing the similarity between classsets. The experiments shows that the minimum support of 1% is good for both accuracy and

speed.

5.3 The result on the ACM data set

The error on the ACM data set is shown in Figure 4. The error is measured over 3197 testing documents. The comparison is consistent with that using the IBM Patent data. The best accuracy of Coverage is typically 30% to 50% higher than of that of kNN.

6 Conclusion

In real life, the class space of documents is a specific-to-general hierarchy and a document may belong to more than one class in the hierarchy. In this paper, an automatic classification of documents with this feature was proposed. In this setting, classes are no longer independent of each other in that they classify some documents in common, and those that classify more documents in common should be considered as more similar to each other than those that classify few documents in common. A notion of similarity of classsets based on the similarity of the documents classified by classsets was proposed to capture this reality. An algorithm for constructing a classifier based on this notion of class similarity was presented. Experiments on real life datasets show that the proposed method achieves much higher accuracy than traditional classifiers.

Bibliography

[1] R. Agrawal, T. Imielinski, and A. Swami, *Mining Association Rules between Sets of Items in Large Databases*, SIGMOD 1993

[2] R. Agrawal and R. Srikant, *Fast Algorithms for Mining Association Rules*, VLDB 1994

[3] S. Chakrabarti, D. Dom, R. Agrawal, and P. Raghavan, *Using Taxonomy, Discriminants, and Signatures for Navigating in Text Databases*, VLDB 1997.

[4] D. Koller and M. Sahami, *Hierarchically Classifying Documents Using Very Few Words*, International Conference on Machine Learning, 1997

[5] J.R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, 1993

[6] H. Schutze, D.A. Hull, and J.O. Pederson, *A Comparison of Classifiers and Document Representations for the Routing Problem*, SIGIR 1995, 229-237

[7] K. Wang, S. Zhou, S.C. Liew, *Building Hierarchical Classifiers Using Class Proximity*, VLDB 1999

[8] Y. Yang and C.G. Chute, *A Linear Least Squares Fit Mapping Method for Information Retrieval from Natural Language Texts*, COLING-92, 1992

[9] Y. Yang and J.O. Pederson, *A Comparative Study on Feature Selection in Text Categorization*, International Conference on Machine Learning 1997.

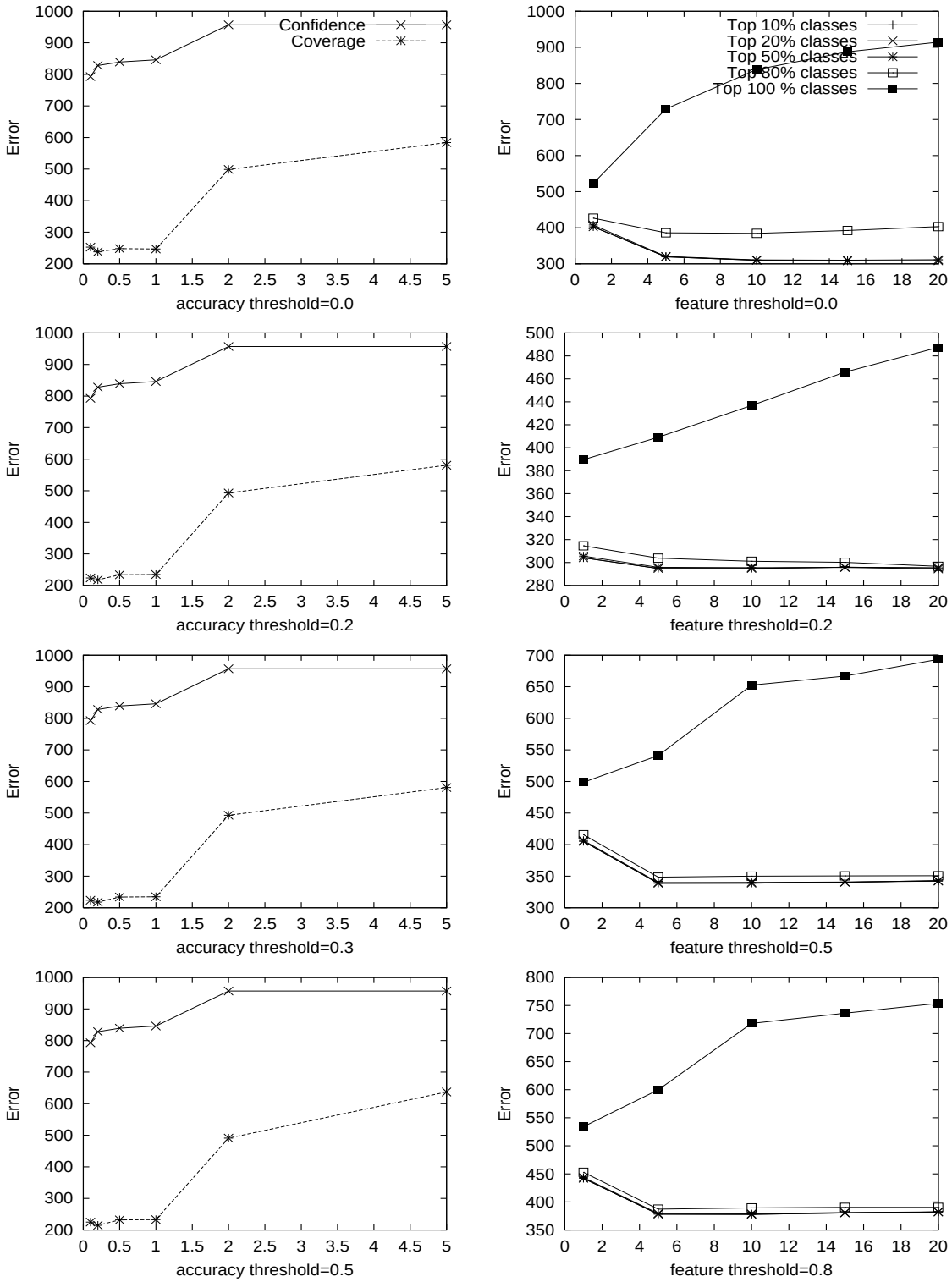


Figure 2. *Patent data set:* Coverage *and* Confidence *(left) and* kNN *(right)*

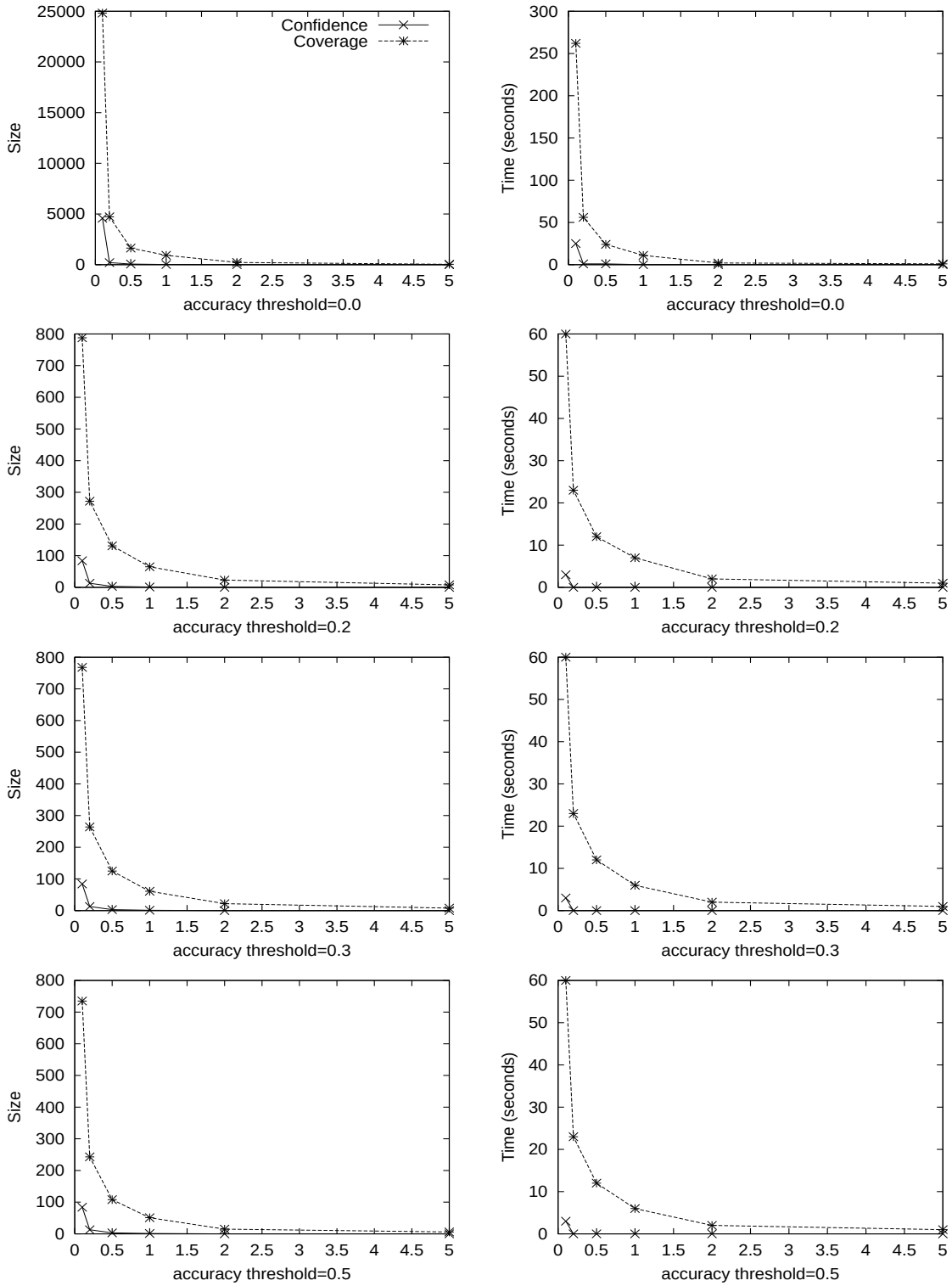


Figure 3. *Patent data set: classifier size (left) and execution time*

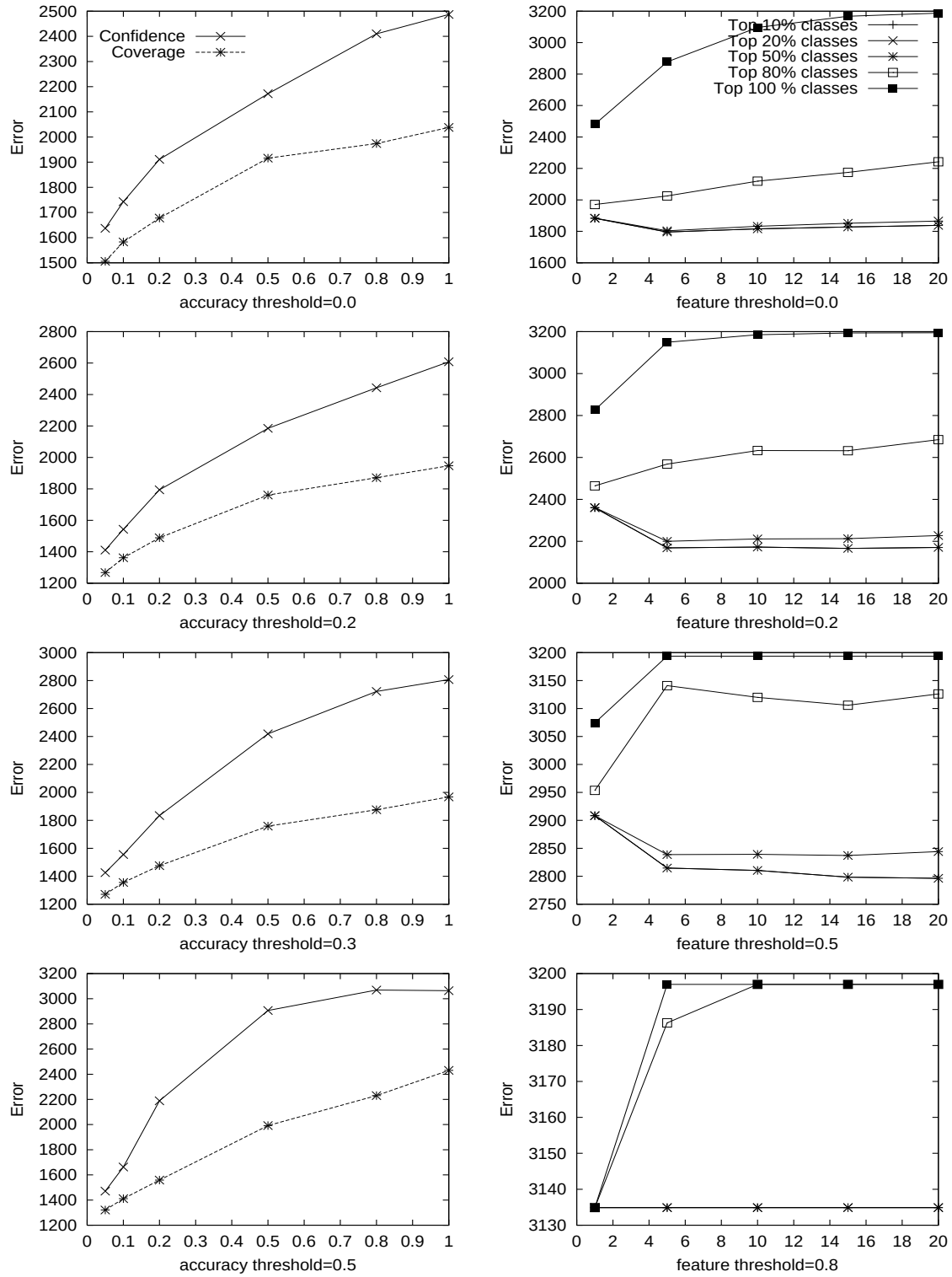


Figure 4. *ACM data set:* Coverage *and* Confidence *(left) and* kNN *(right)*