Building Hierarchical Classifiers Using Class Proximity

Ke Wang Senqiang Zhou Shiang Chen Liew

School of Computing
National University of Singapore
{wangk,zhousenq,liewshia}@comp.nus.edu.sg

Abstract

In this paper, we address the need to automatically classify text documents into topic hierarchies like those in ACM Digital Library and Yahoo!. The existing local approach constructs a classifier at each split of the topic hierarchy. However, the local approach does not address the closeness of classification in hierarchical classification where the concern often is how close a classification is, rather than simply correct or wrong. Also, the local approach puts its bet on classification at higher levels where the classification structure often diminishes. To address these issues, we propose the notion of *class proximity* and cast the hierarchical classification as a flat classification with the class proximity modeling the closeness of classes. Our approach is global in that it constructs a single classifier based on the global information about all classes and class proximity. We leverage generalized association rules as the rule/feature space to address several other issues in hierarchical classification.

1 Introduction

The most successful paradigm for making the mass of information on the Internet comprehensible to every one is by classifying them into topics of hierarchical specificity. Hierarchical classification of this kind has been used in collections of IBM's patent documents (http://www.ibm.com/patents), Library of Congress Catalogue, botanical and animal classification, and Internet search engines such as Yahoo! (http://www.yahoo.com/) and Infoseek (http://infoseek.go.com/) that categorize the content of the World Wide Web. Other applications of hierarchical classification are building directories, bookmarks, email folders, product catalogs, etc. Briefly, in *hierarchical classification*, each training document is a set of terms (i.e., words or phrases) and is labeled by one class (i.e., the topic), where classes are organized by their specificities into an *is-a* hierarchy [1] (i.e., a taxonomy of classes). The task is to construct a classifier that is able to assign classes to new documents within a "small error". As online documents grow in number and size, automatic hierarchical classification becomes a pressing need. This paper examines issues involved in this automation and proposes solutions to them.

1.1 The issues

The central issue in document classification is separating *feature* terms that determine the classes of documents from *noise* terms that do not. In the context of hierarchical classification, it was observed that this separation depends on the current location in the class hierarchy [CDAR97, KS97]. An example in [CDAR97] is that "car" and "auto" may be good features at the top level of Yahoo!, but become noises when drilled down to *Recreation* : *Automotive*. To address this context-sensitivity, [CDAR97, KS97] determine feature terms and construct a classifier at each split of the class hierarchy. This approach is *local* in that the construction at each split is based on the local information at that split. However, local approaches do not address some important issues.

I. Bias of misclassification. In hierarchical classification, the concern often is how close a classification is, rather than simply correct or wrong: misclassification into a remote class (e.g., a nephew class) incurs a larger error than into a nearby class (e.g., a sibling

Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, 1999.

[1] In this paper, a hierarchy is any directed acyclic graph.

class); misclassification at a higher level (e.g., from *Science* to *Recreation*) incurs a larger error than at a lower level (e.g., from *Track_Cycling* to *Unicycling*); misclassification from a general class into a specific class (e.g., from *Recreation* to *Recreation : Sports*) incurs a larger error than the other way around. The traditional counting of misclassifications like the *confusion matrix* fails to address this closeness of classification.

II. Target-sensitivity of features. Feature terms should be determined with respect to the target class that they characterize. For example, "car" and "auto" may characterize the target class *Recreation : Automotive* but not the target class *Recreation*. We call this the *target-sensitivity*. In comparison, the context-sensitivity in [CDAR97, KS97] addresses the ability of *discriminating the subclasses* at location C, whereas the target-sensitivity addresses the ability of *characterizing the target class* C itself. Indeed, [CDAR97, KS97] score a feature without involving a target class. The lack of target classes often yields weak and non-understandable features.

III. High level structure diminishing. The local approach puts its bet on classification at higher levels, in that errors made at higher levels are not recoverable at lower levels. On the other hand, higher levels are often where the classification structure diminishes, due to the divergence of topics. As mentioned above, features like "car" and "auto" that characterize the lower class *Recreation : Automotive* may not characterize the higher class *Recreation*. Consequently, the local approach makes critical decisions (i.e., those at higher levels) based on less reliable information.

IV. Appropriateness of feature spaces. Traditionally, terms (or variables) are considered one at a time in search for features (e.g., information gain [Q93], fisher index [CDAR97], naive Bayes model [KS97], mutual information and χ^2 statistic [YP97]), and co-occurred terms, which are prevailing in document classification, have not been given the first-class consideration. Also, as classification at higher levels is considered, more general terms need to be explored to discover the classification structure. For example, when going up to *Recreation* in Yahoo!, documents may not share specific terms "reading" and "car", but may share general concepts "indoor" and "outdoor".

V. Understandability of classifiers. The local approach needs to make multiple classifications in a row to classify a document. It is difficult to understand the characteristics of a class from multiple classifications. Furthermore, the features at location C in [CDAR97, KS97] are not the characteristics of C, as explained above. In fact, [CDAR97, KS97] has to use the Bernoulli model to tell the class of a given document. In many applications, it is more desirable to tell the characteristics of a class than to tell the class of a given document. Automatic annotation of document clusters by salient keywords is such an example.

The focus of this paper is to address these issues.

1.2 Our approach

First, we introduce *class proximity* to model the closeness of classification. Then, we cast hierarchical classification as non-hierarchical classification where the class proximity models the bias introduced by class specificity. Our approach is *global* in that it constructs a single classifier based on the global information about classes and class proximity. This addresses issues I, III, V. To address issue IV, correlated features at different levels of abstraction will be searched, and a straightforward method cannot deal with the amount of work required. We incorporate an *is-a* hierarchy of terms and leverage generalized association rules [HF95, SA95] as the rule/feature space. A rule/feature has the form $X \rightarrow C$, where X is a set of terms and C is a target class; in a sense, feature X is "owned" by target class C. This approach can generate *all* correlated features at *all* abstraction levels for a large corpus, by benefiting from the work on association rules [HF95, SA95]. This addresses issues II and IV.

To construct a good classifier, however, a crucial step is to rank rules/features with respect to the classification goal, taking into account class proximity and interaction of rules/features (e.g., redundancy and preference of rules/features). We propose two ranking criteria for this purpose. We present an algorithm for selecting a "good" set of rules/features from generalized association rules in one scan of the documents.

Section 2 presents an overview of our approach. Section 3 defines two ranking criteria of rules. Section 4 presents the classifier construction. Section 5 reports the evaluation result. Section 6 remarks on related work and concludes the paper.

2 The overview

This section gives background information about association rules, defines the problem being studied, and outlines our approach.

2.1 Association rules

The problem of mining *association rules* was first studied in [AIS93] in the context of discovering purchase patterns. Let $\mathcal{I} = \{i_1, i_2, \ldots, i_m\}$ be a set of literals, called items. Let $\mathcal{D}$ be a set of transactions, where each transaction T has a unique identifier and is a set of items such that $T \subseteq \mathcal{I}$. A transaction T *contains* an itemset X (i.e., a set of some items in $\mathcal{I}$) if $X \subseteq T$. The *support* of an itemset X, denoted $sup(X)$, is the number of transactions that contain X. An *association rule* has the form $X \rightarrow Y$, where $X \subset \mathcal{I}$, $Y \subset \mathcal{I}$, and $X \cap Y = \emptyset$. The *support* of association rule $X \rightarrow Y$ is

$sup(XY)$. [2] The *confidence* of association rule $X \rightarrow Y$ is $sup(XY)/sup(X)$. The problem of mining association rules is to generate all association rules that have support and confidence greater than the user-specified minimum support and minimum confidence.

Association rules were extended to the item space organized into an *is-a* hierarchy in [HF95, SA95], where ancestors (e.g., "clothes") are more general than descendants (e.g., "jacket"). If an item is bought in a transaction, all its ancestors are considered bought in the transaction too. To take this effect into account, the support of an itemset is modified as follows. Let T be a transaction and $Anc(T)$ be the set of items in T plus all their ancestors. The *support* of an itemset X is the number of transactions T such that $X \subseteq Anc(T)$. With these modifications, a *generalized association rule* (or *multi-level association rule* in [HF95]) $X \rightarrow Y$ could hold between itemsets X and Y with items from any levels.

2.2 Hierarchical classification

In *hierarchical classification*, we are given: (a) A collection of terms $\mathcal{T}$ (i.e., words or phrases), organized into an *is-a* hierarchy called the *term hierarchy.* (b) A collection of classes $\mathcal{C}$ (i.e., topics), organized into an *is-a* hierarchy called the *class hierarchy.* $\mathcal{T} \cap \mathcal{C} = \emptyset$. (c) A collection of documents $\mathcal{D}$. Each document contains at least one term and exactly one class. Terms and the class in a document can be a non-leaf node in their hierarchies. (d) The *class proximity* $\mathcal{B}(C_i, C_j)$, representing the error made by misclassification from class C_i into class C_j. $\mathcal{B}(C_i, C_j) > 1$ (resp. $\mathcal{B}(C_i, C_j) < 1$) means an error larger than (resp. smaller than) an "usual" misclassification. $\mathcal{B}(C_i, C_i) = 0$ for all classes C_i. The task is to find a set of rules, called a *classifier*, that determines the classes for new documents within a small error. In the traditional classification setting, the term hierarchy and class hierarchy contain only leaf nodes and $\mathcal{B}(C_i, C_j) = 1$ for distinct classes C_i and C_j.

Remarks. The quantitative choice of $\mathcal{B}(C_i, C_j)$, essentially a closeness measure of two members in a family hierarchy, is largely application-dependent. Among others, a natural choice is the shortest distance from C_i to C_j in the class hierarchy, which is the default choice in this paper. In this case the classification problem amounts to minimizing the traversal distance between the true class and the predicted class. In the following discussion, we assume that $\mathcal{B}(C_i, C_j)$ is given as part of the problem specification.

2.3 An optimal classifier

We define some properties to be satisfied by our classifiers. Let X be a set of terms. Let $Anc(X)$ denote the set of terms in X plus all their ancestor terms. Consider classification rule $X \rightarrow C$ and document d. We say that $X \rightarrow C$ *covers* d if $X \subseteq Anc(d)$. We say that $X \rightarrow C$ *classifies* d in a classifier if $X \rightarrow C$ covers d and is used to determine the class of d in that classifier. While several rules may cover d, only one rule can classify d. We say $X \rightarrow C$ covers or classifies d *correctly* (resp. *wrongly*) if C is identical (resp. not identical) to the class of d.

We shall construct the classifier by selecting a number of generalized association rules to optimize the classification goal. The optimality is defined with respect to a given rule ranking criterion. Some rule ranking criteria will be discussed in Section 3. Given a rule ranking criterion, we like to the following principles to be enforced on any classifier.

Classification Principle. Each document is classified either by a selected rule of highest possible rank, or by some default class. This ensures the best classification of each document as per the rule ranking criterion used.

Selection Principle. A rule is selected if and only if it covers at least one document correctly and no selected rule of higher rank covers that document. This ensures the compactness of the classifier in that every selected rule classifies some document correctly.

An algorithm for selecting the rules according to these principles will be presented in Section 4.

An optimal classifier. Let $Rulelist_s$ be the list of selected rules, ordered by the rule ranking criterion. Let L be any prefix of $Rulelist_s$. The *error* of a rule R in L is the error made by R on the documents that R classifies. The *cutoff error* of L is the sum of the errors of all rules in L plus the error made by the default class for L. The *default class* for L, chosen from the classes of the documents not classified by L, is to minimize the error made by classifying these documents into it. An *optimal classifier* is the shortest prefix L that has the minimum cutoff error. (E.g., $<>$, $< a >$, $< a, b >$, $< a, b, c >$, and $< a, b, c, d >$ are prefixes of $< a, b, c, d >$, but $< b >$ and $< a, c >$ are not.) An example of optimal classifiers is given in Section 4.2.

2.4 The outline of construction

Given a rule ranking criterion, we shall construct an optimal classifier in three steps. Step 1 generates all generalized association rules $X \rightarrow C$, where X is a set of terms and C is a class, that satisfy the minimum support and (an optional) minimum ranking criterion specified by the user. This step is similar to mining generalized association rules in [SA95]. However, unlike [SA95], we do not generalize the classes of documents because we are aimed at prediction of classes, and we use a minimum value on the chosen rule ranking criterion instead of the minimum confidence. Step 2 sorts all rules found in Step 1 according to the rule ranking criterion. Step 3 finds the list of selected rules $Rulelist_s$ and computes the cutoff error of every prefix

[2] XY is the shorthand of $X \cup Y$.

of *Rulelist*$_s$. The shortest prefix of *Rulelist*$_s$ that has the minimum cutoff error is returned. The rest of the paper focuses on Steps 2 and 3.

3 Ranking rules/features

Each rule $X \rightarrow C$ found in Step 1 can be considered as feature X for the target class C. Intuitively, X is a good feature for C if it occurs in many documents from class C and few documents from classes that are dissimilar to C. We propose two ranking criteria, with one emphasizing the accuracy of classifiers, and the other emphasizing both accuracy and simplicity of classifiers. Let $X \rightarrow C$ be an association rule. $p(X)$ denotes $sup(X)/|\mathcal{D}|$, $p(XC)$ denotes $sup(XC)/|\mathcal{D}|$, and $p(C|X)$ denotes $p(XC)/p(X)$, where $|\mathcal{D}|$ denotes the number of documents in the document collection $\mathcal{D}$.

3.1 The biased confidence

A natural ranking criterion that emphasizes the classification accuracy is the confidence of rules. Taking the class proximity into account, the *biased confidence*, written as $Conf_{\mathcal{B}}(X \rightarrow C)$, is defined as

$$\frac{p(XC)}{p(XC) + \sum_{C_j \neq C} \mathcal{B}(C_j, C) p(XC_j)} \tag{1}$$

In other words, the frequency of misclassifying C_j into C is weighed by the error $\mathcal{B}(C_j, C)$. The further the class C_j is from the predicted class C, the less confident the rule is. Note that $Conf_{\mathcal{B}}(X \rightarrow C)$ is in $[0, 1]$ and that if $\mathcal{B}(C_j, C) = 1$ for all $C_j \neq C$, $Conf_{\mathcal{B}}(X \rightarrow C)$ degenerates into the usual confidence $Conf(X \rightarrow C) = p(XC)/p(X)$.

3.2 The biased J-measure

The second ranking criterion is a modification of the information-motivated J-measure [SG92]. The standard J-measure of rule $X \rightarrow C$, written as $J(X \rightarrow C)$, is

$$p(X)[p(C|X)\log_2 \frac{p(C|X)}{p(C)} + p(\neg C|X)\log_2 \frac{p(\neg C|X)}{p(\neg C)}]$$

The first term $p(X)$ measures the simplicity of the rule. The term inside the square bracket measures the difference between the posteriori $p(C|X)$ and the priori $p(C)$, thus, the discriminating power of X on the target class C: it has a large value if X has either a positive impact on C, where $p(C|X)$ is larger than $p(C)$, or a negative impact on C, where $p(C|X)$ is smaller than $p(C)$.

To suit our purpose, however, we need to make two modifications to the standard J-measure. First, different non-target classes $C_j \neq C$ need to be distinguished because they have different biases towards the target class C. Second, we like to favor the positive impact of X on C and the negative impact of X on non-target classes C_j; we can do this by replacing $+$ sign for non-target classes with $-$ sign. These modifications yield the *biased J-measure*, written as $J_{\mathcal{B}}(X \rightarrow C)$, defined by

$$p(X)[p(C|X)\log_2 \frac{p(C|X)}{p(C)} - \sum_{C_j \neq C} \mathcal{B}(C_j, C) p(C_j|X) \log_2 \frac{p(C_j|X)}{p(C_j)}] \tag{2}$$

We expect that $J_{\mathcal{B}}$ yields a smaller, thus more understandable classifier than $Conf_{\mathcal{B}}$ because it takes into account both simplicity and discriminating power of a rule.

4 Constructing an optimal classifier

We assume that one of the ranking criteria in Equations (1) and (2) is used. Let *Rulelist* be the list of generalized association rules found in Step 1, ranked by the chosen ranking criterion. We construct an optimal classifier by selecting rules from *Rulelist* according to Selection Principle and Classification Principle in Section 2. First, we state two strategies to prune some rules never selected by these principles. Consider two rules $X_1 \rightarrow C_1$ and $X_2 \rightarrow C_2$. We like to characterize the condition that whenever $X_2 \rightarrow C_2$ covers a document, $X_1 \rightarrow C_1$ covers that document, that is, X_1 is more general than X_2. We denote this condition by $X_1 \preceq X_2$. The following theorem gives a test of $X_1 \preceq X_2$, whose proof is straightforward.

Theorem 1 *$X_1 \preceq X_2$ if and only if $X_1 \subseteq Anc(X_2)$.*

Pruning Strategy 1 below says that if a general rule is ranked higher than a special rule and if both rules have the same target class, the special rule is never selected. Before constructing a classifier, we can apply Pruning Strategy 1 to prune rules.

Pruning Strategy 1 *Assume that $X_1 \rightarrow C$ proceeds $X_2 \rightarrow C$ in Rulelist. If $X_1 \preceq X_2$, $X_2 \rightarrow C$ will not be selected. (Proof in [WZL99].)*

Strategy 2 below says that if a general rule is ranked higher than a special rule and is selected, the special rule is never selected. After selecting a rule, we can apply Pruning Strategy 2 to prune other rules.

Pruning Strategy 2 *Assume that $X_1 \rightarrow C_1$ proceeds $X_2 \rightarrow C_2$ in Rulelist. If $X_1 \preceq X_2$ and $X_1 \rightarrow C_1$ is selected, $X_2 \rightarrow C_2$ will not be selected. (Proof in [WZL99].)*

Our construction makes one scan of the documents and keeps track of how each rule in *Rulelist* classifies documents. This information for each rule R is kept in *R.Clist* and *R.Wlist*, which contain the $(id, Class)$ pairs for the documents classified by R correctly and wrongly, respectively. Before all the documents are examined, however, we do not know whether R has the chance to classify a document, as governed by Classification Principle. We adopt a simple strategy: if R is a candidate to classify the current document d, we add $(id, Class)$ for d to *R.Clist* or *R.Wlist*; we prune the $(id, Class)$ pair from *R.Clist* or *R.Wlist* as it becomes known that R has no chance to classify document d.

To illustrate the point, consider two rules $R1$ and $R2$ such that $R1$ proceeds $R2$ in *Rulelist*. Assume that neither rule is selected and that some $(id, Class)$ is contained in *Clist* or *Wlist* of both rules. Now the next document id' is examined. Suppose that $R1$ covers document id' correctly and no selected rule of higher rank covers the document. By Selection Principle $R1$ is selected, and by Classification Principle $R1$ is deemed to classify document *id*. We now know that $R2$ has no chance to classify document *id*, so we can prune $(id, Class)$ from *Clist* or *Wlist* of $R2$.

The construction has two phases. Phase 1 scans the documents and maintains *Wlist* and *Clist* of the rules involved. Phase 2 makes selection decisions for those rules not yet selected and compute the cutoff error at each selected rule.

4.1 Phase 1: Scan the database

This phase, shown in Figure 1(a), scans the documents and marks a rule in *Rulelist* once it is known that the rule will be selected according to Selection Principle. For the current document d, we find the first rule R in *Rulelist* that covers d. There are two cases, depending on whether R is marked.

Case 1 (lines 30-70): R is marked. R will classify d. So we add the $(id, Class)$ pair of d to *R.Clist* (line 50) or *R.Wlist* (line 70).

Case 2 (lines 80-210): R is not marked. There are two subcases, depending on whether R covers d correctly.

- *Case 2a* (lines 90-110): R covers d correctly. In this case R will be selected according to Selection Principle. We mark R and add the $(id, Class)$ pair of d to *R.Clist* (lines 100-110). The marking of R, denoted by $Mark(R)$, include the following steps: delete all $(id, Class)$ pairs in *R.Clist* or *R.Wlist* from all covering rules of d because they do not have the chance to classify these documents, and apply Pruning Strategy 2 to prune more rules. These implementations will be discussed below.
- *Case 2b* (lines 120-210): R covers d wrongly. Since R has not been marked, all rules that cover d (which must be after R) are candidates for classifying d. So, we add $(id, Class)$ of d to the *Clist* or *Wlist* of these rules. However, it is not necessary to consider all such rules: we can stop as soon as any of these rules, say R', was already marked or covers d correctly, whichever comes first in *Rulelist*. The reason is that, by Classification Principle, all rules that come after R' have no chance to classify d. These steps are given in lines 130-210. To simplify the presentation, we assume that a dummy rule at the end of *Rulelist* cover all documents.

Implementation details. At lines 20 and 150, we need to find the rules that cover document d. This operation is similar to the subset function of finding the candidate itemsets contained in $Anc(d)$, implemented by the *hash-tree* in [AIS93, AS94]. For our purpose, we store all rules $X \rightarrow C$ in the hash-tree by treating X as an itemset. Then finding all covering rules $X \rightarrow C$ of document d amounts to finding all itemsets X such that $X \subseteq Anc(d)$. Another implementation concerns with $Mark(R)$ (and line 100 in Phase 2 below) where we need to delete a given $(id, Class)$ pair from the rules whose *Clist* or *Wlist* contain the pair. To locate these rules quickly, as a new $(id, Class)$ pair is added, we can chain up the entries for $(id, Class)$ in the order of the rules involved. To delete a $(id, Class)$ pair, we simply scan the chain for the pair and delete its entry from each rule encountered. To delete more than one $(id, Class)$ pair, we combine their scans and delete their entries in one scan.

4.2 Phase 2: Select final rules

Phase 2, shown in Figure 1(b), scans *Rulelist* to select rules and determine the best cutoff point. Consider the current rule R. There are two cases, depending on whether R is marked.

Case 1 (lines 20-60): R is marked. We append R to $Rulelist_s$ (which is initially empty) and remove R from *Rulelist* (line 30). $Remain[C]$ denotes the number of documents in class C that have not been classified by $Rulelist_s$. $Remain[C]$ is updated to reflect that the documents in *R.Clist* and *R.Wlist* are now classified by $Rulelist_s$ (lines 40-50). Also, we compute the cutoff error of $Rulelist_s$, done in $CutoffError(R)$ (line 60). The cutoff error is defined as $R.RE + R.DE$, where $R.RE$ is the total error of the rules in $Rulelist_s$ and $R.DE$ is the default error of using some default class on the documents not classified by $Rulelist_s$. The class that minimizes the default error is chosen as the default class, denoted $R.DC$.

Case 2 (lines 80-120): R is not marked. In this case, *R.Clist* must be empty, as shown in Lemma 1 below. Therefore, we simply remove R from *Rulelist* (line 80). Now R is no longer a candidate to classify the documents in *R.Wlist*, which triggers the marking of more rules (lines 90-120): for each $(id, Class)$ in *R.Wlist*, we find the first rule R' such that $R'.Clist$ or $R'.Wlist$ contains $(id, Class)$. If R' is not found, document *id* will be classified by a default class. If R' is found, we check whether R' covers document *id*

```
Phase 1:
for each document $d$ do
    find the first rule $R$ in $Rulelist$ that covers $d$;
    if $R$ is marked then /* Case 1 */
        if $R$ covers $d$ correctly then
            add $(id, Class)$ of $d$ to $R.Clist$;
        else
            add $(id, Class)$ of $d$ to $R.Wlist$;
    else /* Case 2 */
        if $R$ covers $d$ correctly then /* Case 2a */
            add $(id, Class)$ of $d$ to $R.Clist$;
            $Mark(R)$;
        else /* Case 2b */
            repeat
                add $(id, Class)$ of $d$ to $R.Wlist$;
                $R$=the next rule covering $d$;
            until $R$ is the dummy rule, or $R$ covers $d$ correctly, or $R$ is marked;
            if $R$ is not the dummy rule then
                if $R$ covers $d$ correctly then
                    add $(id, Class)$ of $d$ to $R.Clist$;
                else if $R$ is marked then
                    add $(id, Class)$ of $d$ to $R.Wlist$;
```

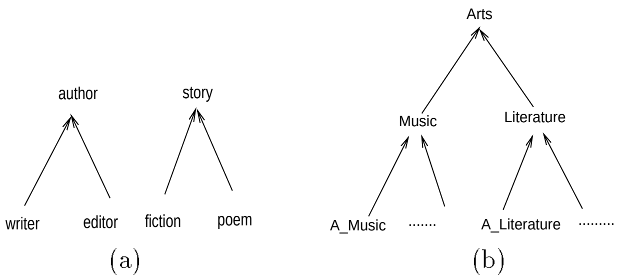
(a)

```
Phase 2:
    $RE = 0$;
for each rule $R$ in $Rulelist$ in the ranked order do
    if $R$ is marked then /* Case 1 */
        append $R$ to $Rulelist_s$ and delete $R$ from $Rulelist$;
        for each $(id, Class) \in R.Clist \cup R.Wlist$ do
            $Remain[Class] = Remain[Class] - 1$;
        $CutoffError(R)$; /* compute the cutoff error at $R$ */
    else /* Case 2 */
        delete $R$ from $Rulelist$;
        for each $(id, Class)$ in $R.Wlist$ do
            find the first rule $R'$ in $Rulelist$ such that $(id, Class)$ is in $R'.Clist \cup R'.Wlist$;
            if $R'$ is found then
                if $(id, Class)$ is in $R'.Clist$ and $R'$ is not marked yet then $Mark(R')$;
find the first rule $R$ in $Rulelist_s$ that minimizes the cutoff error $R.RE + R.DE$;
return the prefix of $Rulelist_s$ ending at $R$, and the default class $R.DC$;

$CutoffError(R)$:
    $R.DE$ =the maximum machine value;
    for each class $C$ such that $Remain[C] \neq 0$ do
        $x_C = \sum_{C'} Remain[C'] \times \mathcal{B}(C', C)$; /* the default error of using $C$ as the default class */
        if $x_C < R.DE$ then
            $R.DE = x_C$ and $R.DC = C$;
    $RE = RE + \sum_{(id,Class) \in R.Wlist} \mathcal{B}(Class, R.Class)$; /* $R.Class$ denotes the class in $R$ */
    $R.RE = RE$;
```

(b)

Figure 1: Step 3



id	term	class
d1	hall,composer	Music
d2	hall,conductor	Music
d3	hall,States	A_Music
d4	States,book	A_Literature
d5	story,States	A_Literature
d6	hall,fiction,writer	Literature
d7	editor,poem	Literature

(c)

Figure 2: Hierarchical classification

correctly and is not marked yet. If so, we mark R' by calling $Mark(R')$ (line 120).

Finally, the shortest prefix of $Rulelist_s$ that has the minimum cutoff error is returned as an optimal classifier (lines 130-140).

Lemma 1 *In Phase 2, if the current rule R is not marked, R.Clist is empty. (Proof in [WZL99].)*

The following theorem follows from our construction algorithm.

Theorem 2 *(a) The full list Rulelist$_s$ satisfies Classification Principle and Selection Principle. (b) The prefix of Rulelist$_s$ returned by the algorithm is an optimal classifier.*

Example 1 Consider the example in Figure 2, where (a), (b), and (c) give the term hierarchy, the class hierarchy, and the training documents. Assume that $\mathcal{B}(C_i, C_j)$ measures the shortest distance from C_i to C_j in class hierarchy. Suppose that the minimum support is 2. We consider four search strategies: (), (B), (T), and (B,T). T means that term hierarchy is used and B means that class proximity is used in the chosen ranking criterion. For all strategies, the error of a classifier is computed using class proximity.

() Strategy. The term hierarchy is ignored and the usual confidence $Conf$ is used. Only two rules satisfy the minimum support:

R3: States → A_Literature
    ($Conf$=0.67, $Clist$=d4,d5, $Wlist$=d3(4))
R4: hall D → Music
    ($Conf$=0.50, $Clist$=d1,d2, $Wlist$=d6(2))

The number following each document id in $Wlist$ is the error on that document. For example, the error made on d3 by R3 is 4 because $\mathcal{B}(A\_Music, A\_Literature) = 4$. Both rules are selected because each classifies some documents correctly. To find an optimal classifier, each prefix of $< R3, R4 >$ is considered, shown in Table 1(). For prefix <>, all documents are classified by default class $Literature$, giving the minimum error of 9. For prefix $< R3 >$, the default class for the remaining d1, d2, d6, d7 is either $Music$ or $Literature$, giving the minimum default error of 4. Thus, the cutoff error of $< R3 >$ is 8. Finally, the cutoff error of prefix $< R3, R4 >$ is 6, with default class $Literature$. So $< R3, R4 >$ is an optimal classifier.

(B) Strategy. By considering class proximity, R4 is now ranked higher than R3:

R4: hall → Music
    ($Conf_{\mathcal{B}}$=0.4, $Clist$=d1,d2, $Wlist$=d3(1),d6(2))
R3: States → A_Literature ($Conf_{\mathcal{B}}$=0.33, $Clist$=d4,d5)

Table 1(B) shows the cutoff error for every prefix of $< R4, R3 >$. $< R4, R3 >$ with default class $Literature$ is an optimal classifier, where the cutoff error is 3.

(T) Strategy. By considering term hierarchy, five rules now satisfy the minimum support:

R0: author,story → Literature ($Conf$=1, $Clist$=d6,d7)
R1: author → Literature ($Conf$=1, not selected)
R2: story → Literature
    ($Conf$=0.67, $Wlist$=d5(1), not selected)
R3: States → A_Literature
    ($Conf$=0.67, $Clist$=d4,d5, $Wlist$=d3(4))
R4: hall → Music ($Conf$=0.50, $Clist$=d1,d2)

By Classification Principle, R1 and R2 do not classify any document correctly, so are not selected, by Selection Principle. Table 1(T) shows the cutoff error for each prefix of selected rules $< R0, R3, R4 >$. $< R0, R3 >$ with default class $Music$ is an optimal classifier, where the cutoff error is 4.

(B,T) Strategy. The class proximity changes the relative rank of R3 and R4:

R0: author,story → Literature ($Conf_{\mathcal{B}}$=1, $Clist$=d6,d7)
R1: author → Literature ($Conf_{\mathcal{B}}$=1, not selected)
R2: story → Literature
    ($Conf_{\mathcal{B}}$=0.67, $Wlist$=d5(1), not selected)
R4: hall → Music ($Conf_{\mathcal{B}}$=0.4, $Clist$=d1,d2, $Wlist$=d3(1))
R3: States → A_Literature ($Conf_{\mathcal{B}}$=0.33, $Clist$=d4,d5)

As before, R1 and R2 are not selected. Table 1(B,T) shows the cutoff error for each prefix of $< R0, R4, R3 >$. $< R0, R4 >$ with default class $A\_Literature$ is an optimal classifier. The cutoff error is 1. □

From the four strategies considered, (B,T) produces the classifier with the smallest cutoff error. Comparison of (B) with (), and (B,T) with (T), shows that class proximity helps to rank R3 and R4 in an order that produces a small error. Comparison of (T) with (),

prefix	last rule's error	default class	default error	cutoff error
$<>$	0	Literature	9	9
$< R3 >$	4	Music or Literature	4	8
$< R3, R4 >$	2	Literature	0	6

(): Term hierarchy=off and class proximity=off

prefix	last rule's error	default class	default error	cutoff error
$<>$	0	Literature	9	9
$< R4 >$	3	A_Literature	1	4
$< R4, R3 >$	0	Literature	0	3

(B): Term hierarchy=off and class proximity=on

prefix	last rule's error	default class	default error	cutoff error
$<>$	0	Literature	9	9
$< R0 >$	0	Music	7	7
$< R0, R3 >$	4	Music	0	4
$< R0, R3, R4 >$	0			4

(T): Term hierarchy=on and class proximity=off

prefix	last rule's error	default class	default error	cutoff error
$<>$	0	Literature	9	9
$< R0 >$	0	Music	7	7
$< R0, R4 >$	1	A_Literature	0	1
$< R0, R4, R3 >$	0			1

(B,T): Term hierarchy=on and class proximity=on

Table 1: Four cases of classifier construction

and (B,T) with (B), shows that term hierarchy helps to capture the classification structure at proper concept levels.

5 Experiments

This section evaluates the effectiveness and efficiency of our approach. For effectiveness, we consider the rules, the error, and the size of the classifier constructed. For efficiency, we consider the execution time and the number of document ids kept in memory. To reveal the sources of effectiveness, we consider the following parameters in our approach: the minimum support, class proximity (on or off), term hierarchy (on or off), and ranking criteria (the biased confidence or the biased J-measure). For comparison, we have implemented the fisher index method in [CDAR97, CDI98], a local approach to hierarchical classification by constructing one classifier at each split of the class hierarchy. The traditional classification methods based on a flatten class space are not a good candidate for comparison because they ignore the hierarchical structure of classes. Also, such methods cannot handle tens of thousands of terms, as in our case, because they either assume independence of terms (like the Naive Bayes classification [KS97]) or consider terms one at a time (like decision trees [Q93]). As in Example 1, we use (), (B), (T), (B,T) to represent different search strategies of our approach. (CDAR97,T) and (CDAR97) denote the local approach in [CDAR97] where term hierarchy is turned on and off. All results presented are the averaged result of the 5-fold cross-validation trial [3].

5.1 The data sets

The ACM data set. The ACM Digital Library (http://www.acm.org/dl/toc.html/) is chosen because we can use its classification system to construct both class hierarchy and term hierarchy (see below). Each paper has five logical parts: (a) Title, (b) Categories and Subject Descriptors, (c) General Terms, (d) Abstract, (e) Full Text. Only parts (a) and (b) are compulsory. The classification information is contained in part (b) and is organized into a hierarchy of four levels. An example path in this hierarchy is:

Hardware (B) — level-1 category
Memory_Structure (B.3) — level-2 category
Design_Style (B.3.2) — level-3 category
Cache_Memories — level-4 subject descriptor

Our classification task is determining the level-1 or level-2 category of a paper using Title in part (a) and

[3] In a k-fold cross-validation trial, a data set is partitioned into k buckets of equal size and k runs are performed by using a different bucket each time as the testing set and the remaining buckets as the training set.

subject descriptors in part (b). The level-3 categories are reserved as generalizing concepts of subject descriptors. The fact that the Title part of a paper is chosen by the authors themselves and the subject descriptors are cross-referenced among categories makes the classification task challenging.

The data set was obtained as follows. The class hierarchy consists of the level-1 and level-2 categories. The term hierarchy consists of the level-3 categories and level-4 subject descriptors. For each paper, a document is created to contain its keywords in Title and level-4 subject descriptors in Categories and Subject Descriptors. If the paper has a category of the form X.0 (i.e., the GENERAL subcategory of X), we choose X as the class of the document; otherwise, we choose a majority level-2 category of the paper as the class. After removing the classes with less than 20 documents, we are left with the ACM data set shown in Table 2. The size of training set and testing set is determined by the 5-fold cross-validation trial.

The Sports data set. For the second data set, we choose the *Recreation* : *Sports* hierarchy in Yahoo! (http://dir.yahoo.com/recreation/sports) because its deep class hierarchy well suits the effectiveness study of class proximity. We descend the Sports hierarchy and ignore the classes with less than 20 documents each. Each document corresponds to a page pointed by a link in a Sports page (with the prefix http://dir.yahoo.com/recreation/sports) but outside the Yahoo!'s domain (without the prefix http://dir.yahoo.com/). The document consists of the keywords tagged by this link. We ignore short-cuts and links to non-Sports pages within Yahoo!. This gives us the Sports data set in Table 2. About 90% of the terms occur in no more than 10 documents and many documents contain only such terms. This makes the classification task more challenging than the ACM data set.

For both data sets, the class proximity is the shortest distance between classes in class hierarchy.

number	ACM data	Sports data
documents	26,515	7,550
classes	78	367
terms	14,754	10,747
levels of class hierarchy	2	7
training documents	21,212	6,040
testing documents	5,303	1,510

Table 2: The statistics about data sets

5.2 The result on the ACM data set

The rules/features found. Figure 3 shows a small sample of features found by (CDAR97,T) and rules found by (B,T) (the biased confidence and minimum support of 0.1%). All terms shown are in the processed form where plural and morphological variations are removed by using the standard text processing in IR. For each rule, the first number is the biased confidence and the second is the support.

According to [CDAR97], the features found at location C have a large variance in the subclasses of C, thus, are discriminators of the subclasses. But such features cannot serve as the characteristics of C itself. For example, "visual" appears in 0.55% of the documents under CSO where it was found as a feature, but appears 0.83% of the documents under $Software$ where it was not found as a feature. This is so because "visual" has a large variance in the subclasses of CSO, but not in the subclasses of $Software$. Such features do not fulfill our goal of characterizing CSO. On the other hand, the rules found by (B,T) clearly tell what terms characterize what subclasses of CSO, which is not the case from examining the features at CSO found by (CDAR97,T).

The following discussion refers to Figure 5. The two ranking criteria divide figures into the left column and the right column. The x-axis denotes the minimum support of x% of the training size. The legend in the figure labeled "Size" is uniformly used in all figures.

The error. The two figures labeled "Error" show the total error on the 5,303 testing documents as defined by class proximity. We can see the following points. (a) (B,T) performs the best, in fact, improves upon (CDAT97) and (CDAR97,T) by as much as 75%. (b) Comparing () with (T), (B) with (B,T) reveals that the global approach benefits drastically from term hierarchy, but not much for the local approach. (c) Comparing () with (B), and (T) with (B,T) reveals that class proximity reduces the total error, but only marginally, due to the shallow class hierarchy. (d) The biased confidence yields higher accuracy than the biased J-measure.

Figure 4 shows the distribution of errors according to the distance between the known class and the predicted class, called the *fatalness* of error. The minimum support is 0.1% (of the training size) for our approach. Clearly, (CDAR97) and (CDAR97,T) make far more fatal errors (of distance 3 or 4) than the global approach. Indeed, we observed that 3,624 or 68% and 3,824 or 72% testing documents were wrongly classified at the top level of the class hierarchy by (CDAR97) and (CDAR97,T), respectively, compared to only 21% by (B,T) (the biased confidence). A similar trend was observed for the Sports data set (see below). This confirms the point made in Introduction that high level structures diminish in the local approach.

The count of the usual misclassification is shown in the two figures labeled "Count". We notice that (CDAR97) and (CDAR97,T) make much more misclassifications for the ACM data, i.e., 70%, than for the USPatent data in [CDAR97], i.e., about 25%. This difference is because the ACM data has 78 classes, compared to only 12 classes for the USPatent data.

Best features found by (CDAR97,T):
At Computer_Systems_Organization (CSO):
medium, mainfram, super, attribut, techniqu, comput, stream, multipl, x_mp, embed, apl, train, cyber, oop, win, council, visual, etc.
At Software:
object_oriented_programming, concurrent_programming, classif, processor, featur, techniqu, construct, tool, process, storag, parallel_programming, organiz, compil, file_system, distributed_system, protect, etc.

Best rules found by (B,T):
Under Computer_Systems_Organization (CSO):
vector,stream,processor,parallel → Processor_Architectures (1.00, 22)
multiple_instruction_stream → Processor_Architectures (1.00, 55)
data_flow,architectur → Processor_Architectures (1.00, 30)
internet,architectur → Computer_Communication_Networks (1.00, 67)
mode,atm → Computer_Communication_Networks (1.00, 32)
network,circuit_switching → Computer_Communication_Networks (1.00, 25)
techniqu,model,attribut → Performance_of_Systems (0.94, 65)
Under Software:
program,function,applicative → Programming_Techniques (0.87,52)
object_oriented_programming → Programming_Techniques (0.86,258)
reusable_software → Software_Engineering (0.97,200)
software,methodologie → Software_Engineering (0.92, 55)
programming_environment → Software_Engineering (0.89, 287)
processor,parse → Programming_Languages (1.00, 27)
processor,compiler → Programming_Languages (0.91, 454)
organization,distributed_system → Operating_Systems (1.00, 71)
synchronization,process → Operating_Systems (1.00, 53)

Figure 3: The comparision of features and rules

The size of classifiers. The two figures labeled "Size" show the size of classifiers, which is the number of rules for our approach, and the number of features for (CDAR97,T) and (CDAR97). Clearly, the classifiers produced by (CDAR97,T) and (CDAR97) are much larger, thus, less understandable, than those produced by our approach. The use of term hierarchy has increased the size of classifiers. The biased J-measure yields consistently fewer rules than the biased confidence.

The execution time. The two figures labeled "Time" show the execution time. For our approach, most time was spent on generating association rules. For (CDAR97) and (CDAR97,T), most time was spent on computing the fisher index of terms and determining the cutoff point of the feature list where every prefix of the feature list was examined for each document in the validation set. Our algorithms are much faster than (CDAR97) and (CDAR97,T).

Document ids kept. The two figures labeled "Ids" show the number of document ids kept in *Clist* and *Wlist* in our approach. Recall that the training set has 21,212 documents. Thus, each document id is kept no more than twice. This number drops quickly for a smaller minimum support. We can further reduce this number by keeping *Clist* and *Wlist* only for the rules that are not marked at any time. We omit this detail due to space limitation.

5.3 The result on the Sports data set

For this data set, a similar trend was observed on the size of classifiers, execution time, and number of document ids kept. Also, the remark about the rules and features for the ACM data set is applicable to this data set. The detail can found in [WZL99]. Here we report briefly on the error of classification. As the minimum support varies from 0.02% to 0.5%, the total error of (B) ranges from 2700 to 3700, much smaller than the total error of (CDAR97), which is 5700, and the total error of (), which ranges from 3300 to 5800. (Note that the Sports data set has no term hierarchy.) Again, we observed the trend that (CDAR97) and () more frequently make fatal errors than (B). In fact, 68% of the testing documents were classified wrongly at the top level by (CDAR97), compared to only 37% by (B) (the biased confidence and minimum support of 0.2%). This shows that the global approach based on class proximity indeed achieves the closeness of classification.

6 Concluding remarks

With few exceptions, most work on (supervised) classification ignored the structure of features and classes, e.g., [Q93, SHP95, SOM, YP97]. Recently, hierarchically structured features and hierarchically structured classes were examined in [AAK96] and [CDAR97, KS97], respectively. Related but different topics are

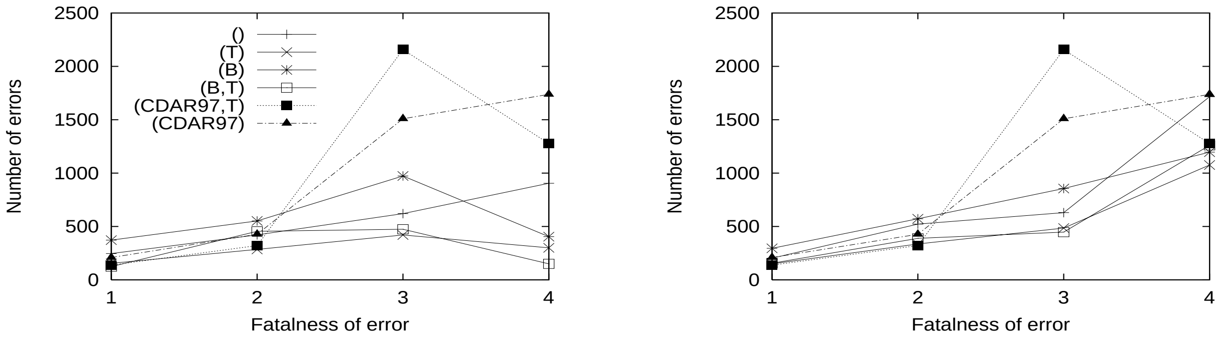


Figure 4: ACM error distribution: biased confidence (left) and biased J-measure (right)

hypertext categorization [CDI98] where some neighbourhood of interconnected documents was explored to enhance the classification accuracy. Association rules [AIS93, AS94, SA95, HF95] were proposed with a different mind set from classification. [LHM98] integrated association rules and classification rules for a relation table, but did not consider hierarchical classification. Also, the algorithm in [LHM98] is rather complex and the database is scanned more than once. Finally, none of these work has considered the notion of class proximity.

This paper makes the following contributions. First, it identifies several important issues in hierarchical classification. Then, it proposes a new approach to hierarchical classification by aiming at the closeness of classification, which is fundamentally different from earlier approaches. The closeness of classification is relevant not only to hierarchical classification, but also to the general setting of classification. For example, classifying *Urgent* emails into *Junk* emails is much more costly than the other way around, and the closeness of classification is useful to minimize misclassification in a way sensible to such applications. Several characteristics make our approach robust and scalable to a large corpus, namely, construction of a global classifier, search for multi-level abstraction and correlation of features, determination of features with respect to target classes, and a single scan of the document database. Experiments have shown encouraging results.

References

[AAK96] H. Almualim, Y. Akiba, S. Kaneda, "An efficient algorithm for finding optimal gain-ratio multiple-split tests on hierarchical attributes in decision tree learning", National Conference on Artificial Intelligence, AAAI 1996, 703-708

[AIS93] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases", SIGMOD 1993, 207-216

[AS94] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules", VLDB 1994, 487-499

[CDAR97] S. Chakrabarti, D. Dom, R. Agrawal, and P. Raghavan, "Using taxonomy, discriminants, and signatures for navigating in text databases", VLDB 1997, 446-455 (Also see: "Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies", The VLDB Journal (1998) Vol. 7, No. 3, 163-178)

[CDI98] S. Chakrabarti, B. Dom, and P. Indyk, "Enhanced hypertext categorization using hyperlinks", SIGMOD 1998, 307-318

[HF95] J. Han and Y. Fu, "Discovery of multiple-level association rules from large databases", VLDB 1995, 420-431

[KS97] D. Koller and M. Sahami, "Hierarchically classifying documents using very few words", International Conference on Machine Learning, 1997, 170-178

[LHM98] B. Liu, W. Hsu, and Y. Ma, "Intergrating classification and association rule mining", KDD 1998, 80-86

[WZL99] K. Wang, S. Zhou, S.C. Liew, "Building hierarchical classifiers using class proximity", Technical Report, National University of Singapore, 1999

[Q93] J.R. Quinlan, C4.5: programs for machine learning, Morgan Kaufmann, 1993

[SA95] R. Srikant and R. Agrawal, "Mining generalized association rules", VLDB 1995, 407-419

[SG92] P. Smyth and R. Goodman, "An information theoretic approach to rule induction from databases", IEEE Transactions on Knowledge and Data Engineering, Vol. 4, No. 4, Aug 1992, 301-316

[SHP95] H. Schutze, D.A. Hull, and J.O. Pederson, "A comparison of classifiers and document representations for the routing problem", SIGIR 1995, 229-237

[SOM] Self-organizing map, http://www.cis.hut.fi/nnrc/nnrc-programs.html

[YP97] Y, Yang and J.O. Pederson, "A comparative study on feature selection in text categorization", International Conference on Machine Learning 1997.

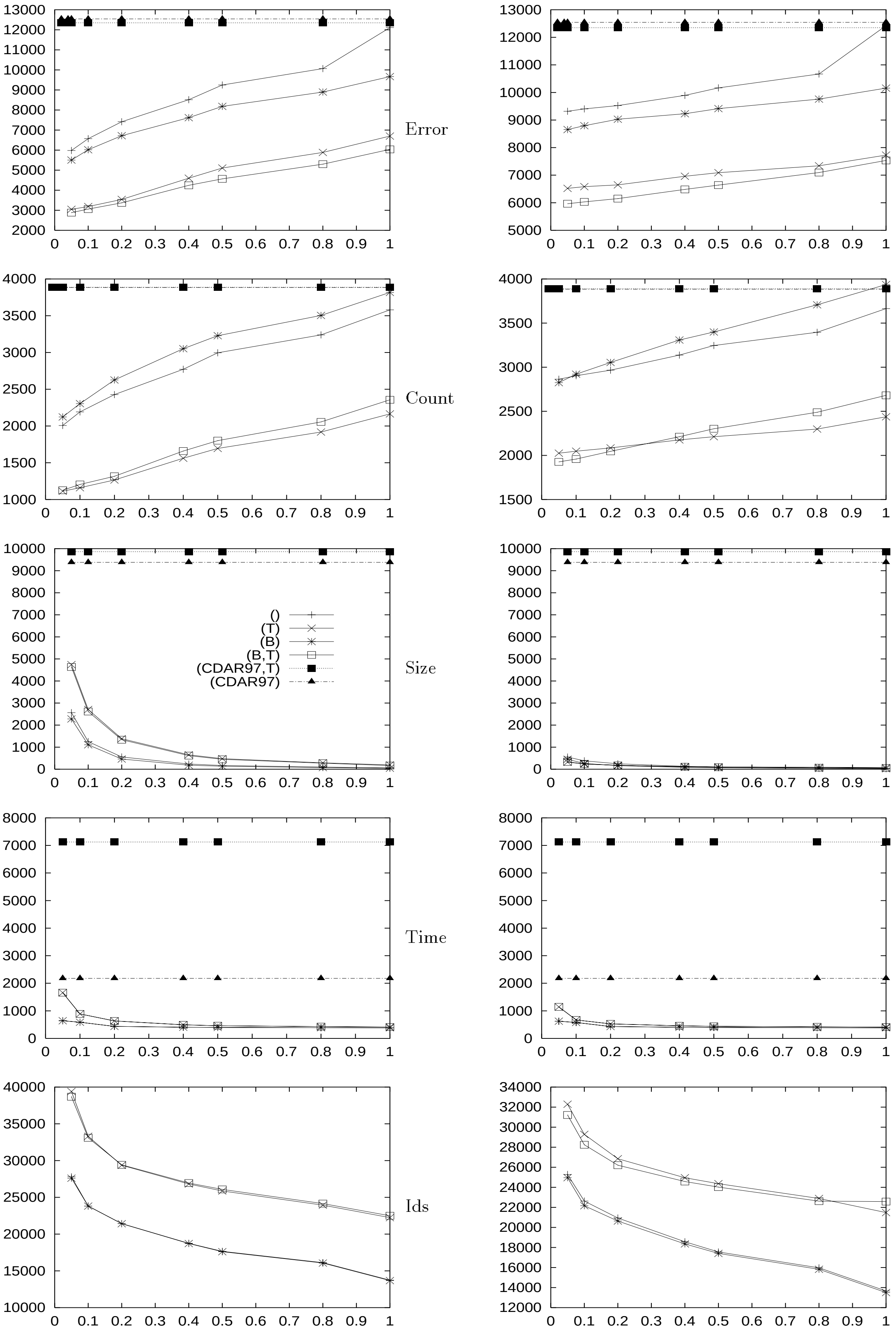


Figure 5: ACM: biased confidence (left) and biased J-measure (right)