On the Strength of Hyperclique Patterns for Text Categorization ⋆

Tieyun Qian [a,*], Hui Xiong [b], Yuanzhen Wang [c], Enhong Chen [d]

[a] *Department of Computer Science, Wuhan University*
Email: qty@whu.edu.cn

[b] *Management Science and Information Systems Department, Rutgers University*
Email: hui@rbs.rutgers.edu

[c] *School of Computer Science, Huazhong University of Science and Technology*
Email: wangyz2005@163.com

[d] *Department of Computer Science, University of Science and Technology of China*
Email: cheneh@ustc.edu.cn

Abstract

The use of association patterns for text categorization has attracted great interest and a variety of useful methods have been developed. However, the key characteristics of pattern-based text categorization remain unclear. Indeed, there are still no concrete answers for the following two questions: Firstly, what kind of association pattern is the best candidate for pattern-based text categorization? Secondly, what is the most desirable way to use patterns for text categorization? In this paper, we focus on answering the above two questions. More specifically, we show that hyperclique patterns are more desirable than frequent patterns for text categorization. Along this line, we develop an algorithm for text categorization using hyperclique patterns. As demonstrated by our experimental results on various real-world text documents, our method provides much better computational performance than state-of-the-art methods while retaining classification accuracy.

Key words: Association Rules, Hyperclique Patterns, Text Categorization

⋆ A Preliminary version of this work has been published as a two-page short paper in ACM CIKM 2006.

* Corresponding author.

1 Introduction

Text categorization is a key technique for processing and organizing text documents. Text categorization techniques are often used to classify news stories and to guide a user's search on the Web. Recently, there has been considerable interest in using association patterns [1] for text categorization [2, 9, 21, 22, 28, 34]. This is known as Associative Text Categorization (ATC). A key benefit of ATC is its ability to produce semantic-aware classifiers which include understandable rules for text categorization. While several promising algorithms have been developed, further investigation is needed to characterize associative text categorization with respect to the following two issues:

(1) What kind of association pattern is the best candidate for associative text categorization?
(2) What is the most desirable way to use association patterns for text categorization?

The goal of this work is to address the above two issues. Previous methods used frequent itemsets (frequent patterns) [1] for text categorization. Recently, Xiong et al. [36, 37] have defined a new pattern for association analysis-the *hyperclique pattern*-that demonstrates a particularly strong connection between the overall similarity of a set of objects and the itemset in which they are involved. Indeed, the hyperclique pattern (HP) is a better candidate for text categorization than frequent itemset (FI). First, the hyperclique pattern includes the items which are strongly related to each other. Second, hyperclique patterns have much better coverage of items (vocabulary) than frequent itemsets, since hyperclique patterns can capture words with very low frequency while existing FI-mining algorithms often break down when the minimum support threshold is set to a low value. Third, the computational cost of finding hyperclique patterns is significantly lower than the cost for finding frequent itemsets [36]. Finally, there are fewer hyperclique patterns than frequent itemsets under the same minimum support threshold. For example, when the minimum support threshold is set to the value of 5% on the WebKB data set, there are 214672 FIs and 116435 HPs respectively. As a result, hyperclique patterns are more manageable for rule extraction in the process of text categorization.

In this paper, we develop a new algorithm for text categorization using hyperclique patterns. One important algorithm design issue is how to select a subset of candidate patterns for the end classifier. For associative text categorization, all rules that satisfy user-specified minimum support and confidence thresholds will be identified. This is the main strength of associative text categorization because a more accurate classifier can be derived from the rules that provide a general description of the data. This strength can also be a drawback

as it is difficult to select useful rules from a huge number of candidate rules. Some rule-pruning techniques must be applied to the discovered association patterns for accurate and efficient classification. A key question in pruning is what criterion is used to determine which rules should be removed. Indeed, there is an important subtlety between the classification accuracy and the generality of a rule. On one hand, if a rule with higher support (more general) but relatively lower confidence (less accurate) is removed from the rule sets, we may suffer from the risk of missing some test examples since there will be no rule to cover these examples. On the other hand, a rule with low confidence increases the risk of making a wrong decision. In our algorithm, we decide to eliminate rules with both low support and low confidence. This is considered a general-to-specific pruning method. However, it is a quite time-consuming operation to judge general-to-specific ordering among different rules. To address this problem, we employ a *vertical pruning* approach, which can reduce the computational cost and retain classification accuracy.

In addition to the rule-pruning methods, we also utilize feature selection in our algorithm. Specifically, we first demonstrate that several commonly used feature selection metrics can be expressed as a function of confidence and support, then we propose a method to integrate feature selection into the rule pruning step. In this way, we can dynamically determine the best feature set from the candidate patterns.

Finally, our experiments on several real-world document data sets show that our method has much better computational performance than state-of-the-art methods while retaining classification accuracy.

Overview The remainder of this paper is organized as follows. Section 2 presents related work. In Section 3, we introduce some basic concepts. We describe some algorithm design issues in Section 4. Section 5 provides experimental results. Finally, in Section 6, we draw conclusions.

2 Related Work

Associative classification was first introduced in [22], and from then on, several associative classification methods were proposed [7, 21, 23, 24, 34, 35]. When associative classification is applied to text categorization, document data sets are stored as transaction data with words as items and each document as a transaction [2, 28].

Recent work in this area is mainly focused on how to prune rules and then build an accurate classifier. Quite often, the popularity and significance of a rule were studied first. Along this line, the pessimistic error rate [22] and some

other methods such as the chi-square test [21, 23] were widely used. In order to construct a rule-based classifier, most of these methods modified sequential covering techniques to determine when to stop adding more rules to the classifier. For instance, instead of using a database coverage strategy to select rules, the HARMONY method proposed in [34] directly mines the final set of classification rules by pushing some pruning techniques into the projection-based frequent itemset mining framework. However, little effort has been focused on the efficiency issue for rule extraction. Indeed, when associative classification is applied to text documents, it is necessary to set minimum support thresholds to some low values for the purpose of getting better coverage of the topics. Most existing methods conducted experiments mainly on the Reuters data set [20]. Text categorization for the Reuters data set is relatively easy since its documents were written by newspaper reporters, and the terms in this data set are quite consistent and specific. As we illustrate in later sections, the classification accuracy does not improve much as the minimum support thresholds for the Reuters data set decrease. This indicates that one could get reasonable results even at high support thresholds. However, most document data sets have different features than the Reuters data set. In most cases, it is critical to set low support thresholds and get a better coverage of the topics, and thus improve the performance of text categorization. To this end, we explore several techniques such as transaction caching, inverse matching, and vertical path pruning, to efficiently extract rules to form the end classifiers.

In addition to exploiting techniques to improve the efficiency of rule extraction, we also evaluate the choice of association patterns. Previous work in [9] and [30] exploited sentential co-occurring words as patterns to construct the end classifiers. Sentence level text classification considers co-occurrence at the level of sentences rather than documents, thus greatly reducing the amount of rules. However, sentence segmentation itself is a difficult task in some applications, and it is also difficult for sentence-level text classification to find sufficient frequent itemsets in a corpus with diverse topics. Actually, current sentence level classifications lose their efficacy on large data sets, such as WebKB [26] and 20NG data sets [18]. In contrast, the main drawback of document level classification is that it is very time-consuming due to two reasons: items are more liable to co-occur in documents than in sentence level, and a document transaction is longer than a sentence transaction which leads to more matching time both in training and classification stages. In this paper, we try to improve both efficiency and the classification performance of associative text categorization by adapting hyperclique patterns [36, 37] as candidate association patterns for capturing co-occurring words at the document level.

Another major issue related to text categorization is the high dimensionality of the input data. Feature extraction and feature selection are two major categories of dimension reduction approaches. The former tries to produce new features via linear or nonlinear transformations, while the latter selects the

most relevant features. A number of feature extraction approaches, such as Principal Component Analysis (PCA) [16], Latent Semantic Indexing (LSI) [6], probabilistic Latent Semantic Indexing (pLSI) [13], Linear Discriminant Analysis (LDA) [25], Independent Component Analysis (ICA) [17], Locally Linear Embedding [33], and Latent Dirichlet Allocation (LDA) [4], have been proposed in the literature. Also, many feature selection criteria, such as Document Frequency(DF), Mutual Information(MI), Information Gain(IG), Chi-Squared(χ^2), Bi-Normal Separation(BNS), Odds Ratio(OR), combined with different classification methods (kNN, LLSF, Rocchio, Naïve Bayes, SVM) [39, 8, 15, 27, 10, 31, 12, 40], are widely used. However, current associative text classification methods either apply no feature selection method [9], or simply do feature selection in a separate preprocess procedure [28, 2, 34]. The drawback of such a method is that the predetermined features may not be frequent items. For instance, the term '*editorial*' and '*committees*' in the *faculty* class of the *WebKB* data set both have very high information gain value, but the support values of these two words are 5.2% and 4.3%, respectively. If we set the minimum support threshold to 10%, these two terms will certainly be removed. These observations lead us to re-examine feature selection methods in the context of associative text classification.

3 Basic Concepts

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items (words), $T = \{t_1, t_2, \ldots, t_l\}$ be a set of transactions (documents), and C be a set of categories $\{c_1, c_2, \ldots, c_n\}$. Each transaction t is a set of items and $t \subseteq I$. One or multiple categories can be associated with each document, depending on the classification task being single or multi-labeled.

Definition 1 *The **local support** of itemset X in category c_i, denoted as $lsupp(X, c_i)$, is the fraction of transactions in category c_i containing X, where $X \subseteq I$ and $c_i \in C$. The absolute local support of X in c_i, denoted as $|lsupp(X, c_i)|$, is the number of transactions in c_i containing X.*

Definition 2 *The **local confidence** of an itemset $\{X, Y\}$ in category c_i is denoted as $lconf((X \Rightarrow Y), c_i)$ and $lconf((X \Rightarrow Y), c_i) = \frac{lsupp(X \bigcup Y, c_i)}{lsupp(X, c_i)}$, where $X \subseteq I$, $Y \subseteq I$, $X \bigcap Y = \phi$, and $lsupp(X \bigcup Y, c_i)$ is the fraction of transactions in category c_i containing both X and Y.*

Definition 3 *If the local support of an itemset X in category c_i is above a user-specified minimum, minsup, i.e., $lsupp(X, c_i) \geq minsup$, then we say that the itemset X is **a frequent itemset** in category c_i.*

Definition 4 *A **hyperclique** pattern [36] is a new type of association pattern*

that contains items that are highly affiliated with each other. By high affiliation, we mean that the presence of an item in a transaction strongly implies the presence of every other item that belongs to the same hyperclique pattern. The h-confidence measure [36] is specifically designed to capture the strength of this association. The h-confidence of an itemset $P = \{i_1, i_2, \ldots, i_m\}$ *in category* c_i*, denoted as* $hconf(P, c_i)$*, is a measure that reflects the overall affinity among items within the itemset in category* c_i*. This measure is defined as* $\min\{(lconf((i_1 \Rightarrow i_2, \ldots, i_m), c_i), lconf((i_2 \Rightarrow i_1, i_3, \ldots, i_m), c_i), \ldots, lconf((i_m \Rightarrow i_1, \ldots, i_{m-1}), c_i)\}$*, where* $lconf$ *is the conventional definition of local confidence as given above.*

Table 1 shows some hyperclique patterns identified from words in the WebKB dataset, which includes articles from various categories such as 'course', 'faculty', 'project', and 'student'. For instance, in this table, the hyperclique pattern {artificial, intelligence} is from the 'course' category. The absolute local support of this pattern is 41, which means that the term 'artificial' and 'intelligence' cover 41 documents among the total 901 documents in the 'course' category. Since the local support of 'artificial' is 5.5%, the local support of 'intelligence' is 5.8%, and the local support of 'artificial, intelligence' is 5.5%, then

$$lconf(artificial \Rightarrow intelligence) = \frac{lsupp(artificial, intelligence)}{lsupp(artificial)} = 100\%$$

$$lconf(intelligence \Rightarrow artificial) = \frac{lsupp(artificial, intelligence)}{lsupp(intelligence)} = 95.3\%$$

Hence, h-conf(artificial, intelligence) = min{95.3%,100%} = 95.3%.

Table 1
Examples of hyperclique patterns

hyperclique patterns	|lsupp|	lsupp	h-conf
{artificial, intelligence}	41	5.5%	95.3%
{computer, science, professor}	576	63.9%	71.6%
{vitae, curriculum}	30	2.3%	69.8%

The local support and h-confidence value of a hyperclique pattern reflect how frequent the pattern occurs in a single class and how closely the items in a pattern are interrelated to each other. In order to judge the goodness of a rule, we give further definitions for the global support, global confidence, and phi correlation coefficient.

Definition 5 *The* ***global support*** *of an itemset* X*,* $gsupp\ (X)$*, is the fraction of transactions of all the categories containing* X*. The absolute global*

support of an itemset X, denoted by $|gsupp(X)|$, is the number of transactions of all the categories containing X.

Definition 6 *The **global confidence** of an association pattern $X \Rightarrow c_i$ is denoted as $gconf(X \Rightarrow c_i)$ and $gconf(X \Rightarrow c_i) = \frac{|lsupp(X,c_i)|}{|gsupp(X)|}$, where $X \subseteq I$, $c_i \in C$, and $|lsupp(X, c_i)|$ and $|gsupp(X)|$ follow the definitions of absolute local support and absolute global support as given above.*

Definition 7 *The **phi correlation coefficient** of an association pattern $X \Rightarrow c_i$ is denoted as $phi(X \Rightarrow c_i)$ and $phi(X \Rightarrow c_i) = \frac{lsupp(X,c_i) - gsupp(X)gsupp(c_i)}{\sqrt{gsupp(X)gsupp(c_i)(1-gsupp(X))}} * \frac{1}{\sqrt{(1-gsupp(c_i))}}$, where $X \subseteq I$, $c_i \in C$, and $gsupp(c_i)$ is the fraction of transactions in category c_i of all the categories.*

Table 2
Examples of candidate rules

candidate rules	lsupp	h-conf	gsupp	gconf
r_1: {dlrs,shares} ⇒ acq	22.7%	43.5%	7.84%	73.8%
r_2: {outstanding,investment} ⇒ acq	4.4%	30.4%	1.04%	97.3%
r_3: {corn,agriculture} ⇒ corn	38.7%	53.0%	1.2%	92.1%
r_4: {day,oil} ⇒ crude	28.3%	31.4%	1.8%	94.8%

Tabel 2 gives some candidate rules in Reuters dataset. Comparing the rule r_1 and r_2 in Table 2, one can find that both the local support and h-confidence value of r_1 is much larger than those of r_2. However, this does not mean that r_1 is a better rule than r_2. Though the term 'dlrs' and 'shares' in r_1 are often mentioned in the category 'acq', they are also frequent in other categories, thus resulting in a low global confidence value. In contrast, r_2 is a powerful rule since its global confidence reaches a very high value of 97.3%.

4 Algorithm Descriptions

In this section, we describe some design issues for associative text categorization. Specifically, we illustrate: 1) techniques for efficient rule evaluation; 2) techniques for efficient rule pruning; 3) a method to integrate feature selection into the rule extraction procedure; and 4) an approach for adaptively predicting test documents.

Generating Candidate Rules. To solve the problem of class discrepancy, it is necessary to mine hypercliques in each category. Modeling each text document as a transaction and each distinct word as an item, we employ the algorithm called Hyperclique Miner in [36, 37] to mine all frequent hypercliques by category. This algorithm utilizes *cross-support* and *anti-monotone* properties of h-confidence [36, 37] to efficiently discover hyperclique patterns for data sets with skewed support distributions, which are quite common in text corpora. Candidate rules are generated by making hyperclique antecedent and class label consequent. For instance, if $\{grain, wheat, export\}$ is a hyperclique pattern in class $'grain'$ in Reuters-21578, then $\{grain, wheat, export\} \Rightarrow grain$ forms a candidate rule.

To evaluate the relationship between the antecedent and the consequent in a general view, we compare rules according to their differential capability. We use global confidence and phi correlation coefficient to evaluate rules. Global confidence is an estimate of the conditional probability of class c_i given the term X. However, the confidence measure could be misleading since it doesn't reflect the strength of implication between the class and the term, so we further test whether X is correlated with c_i by phi correlation coefficient. To calculate these measures efficiently, we first collect all category rules in a global prefix tree, and then we need one pass over all training documents. This is the most time consuming part. Several considerations are taken to achieve high computational efficiency. We first delay matching by storing all transactions in a one bit matrix, then we invert the matching process by using every rule to search the transaction matrix.

Cache transactions: If each transaction is managed individually, every time a transaction is read into memory, it will be counted once. By caching transactions into a bit matrix, multiple transactions can share one count. In this matrix, each column corresponds to an item while each row to a transaction. A bit is set to 1 only when an item (word) corresponding to the column occurs one or more times in a transaction (document) corresponding to the row, otherwise it is set to 0. Note that duplicate documents are removed before they enter into the matrix, and all items except frequent ones can be removed from each transaction to reduce the size of a transaction in addition to the memory overhead. When the counting procedure finishes, the bit matrix is destroyed to save memory.

Inverse matching: During the matching step, for each transaction with average length m, the convenient way is to check its 2^m subset in the prefix tree where all candidates are stored. However, compared to the explosive subsets of transactions, the number of candidate rules is much smaller. In order to count

the matching times of each rule node to all transactions, we first project its corresponding columns in the bit matrix, and then we perform an intersection operation on these columns row by row. The total count of match times is the number of rows where the intersection result is 1.

A significant performance gain can be achieved by adopting the above techniques. Indeed, when the above two strategies are applied to 20NG corpus, about 2-3 times speedup is achieved at the different parameter settings. After collecting the global supports of rule antecedents in all categories, the confidence and phi correlation measures can be computed easily. Rules with confidence and phi correlation lower than the user-specified minimum confidence and minimum phi correlation are removed. It may be difficult to tune parameters. However, in our experience, setting the min-conf and the min-phi to 0.52 and 0.10 respectively is good enough to prune ineffective rules while keeping the accuracy of the end classifier, so both of these parameters are transparent to end-users. The rationale of this pruning is quite straightforward: A selected rule must make an improvement on accuracy. For a rule r: $I \Rightarrow c$, if the confidence is 0.5, the rule's decision is random, whereas if it is less than 0.5, it is worse than random. Hence, a threshold value higher than 0.50 is preferred. Rules either uncorrelated ($\phi = 0$) or negatively correlated ($\phi < 0$) should be removed, and those rules which are of little correlation ($\phi < 0.1$) will make little contribution to later classification and also should be removed.

4.2 Vertical Pruning with General-to-Specific Ordering

For the corpora involving a wide range of topics, as is common in many text classification tasks, it is necessary to set the support threshold lower in order to obtain a higher recall. However, this makes the number of candidate rules larger. In order to obtain fast response to a classification request, it is important to reduce the size of the rule set. Besides evaluating and pruning a *single* rule with confidence and phi coefficient correlation metrics, we further investigate techniques referring to the discrimination properties of *different* rules, and we propose an approach to efficiently remove rules with lower classification power.

Definition 8 *Given two rules r_1: $I_1 \Rightarrow c$, and r_2: $I_2 \Rightarrow c$, r_1 is said to be more general than r_2 and r_2 is said a specific rule of r_1, if and only if I_1 is a subset of I_2. Also, there is a general-to-specific (g-to-s) ordering between r_1 and r_2.*

Although a number of criteria to quantify the value of rules have been suggested, it is still difficult to find a balance point at which both the generality

and the accuracy of a rule set are optimal. In this study, we compare rules in an existing g-to-s ordering and remove more specific but less accurate ones. This pruning strategy is known as a g-to-s pruning strategy and is also used in [2, 9, 21]. However, our method is different from all previous works.

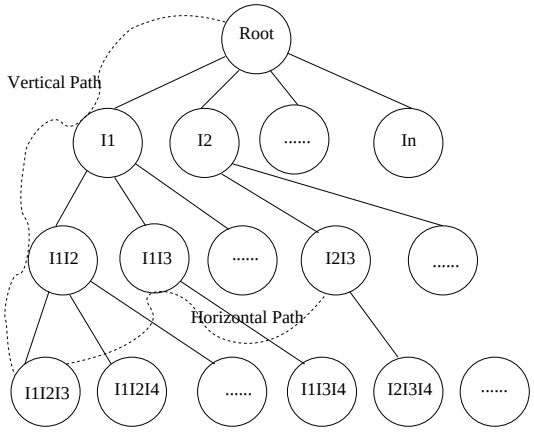


Fig. 1. An example of prefix tree structure

Assuming the association patterns are stored in a prefix tree structure, as shown in Fig.1, not only will rules along the vertical direction possibly exhibit super-subset relationships, but also those along the horizontal direction. For example, in Fig.1, along the vertical path, node I_1 and node I_1I_2 exhibit general-to-specific ordering, and so do node $I_1I_2I_3$ and node I_1I_3 along the horizontal direction.

The vertical check only needs one pass of depth first traversing over the whole tree for all branches, while the horizontal check needs multiple passes traversing over the whole tree for each rule node in the tree. So we only eliminate ineffective specific rules along vertical direction, and we defer the selection step until classification time. If multiple rules exhibiting general-to-specific ordering cover the test instance simultaneously, we only select the best matching rule for the test instance. After the next sequential covering step in the training stage, the number of rules will be greatly reduced, so the testing time will not increase too much. In this way, we aim to get trade-off between the training and testing time.

4.3 Integrating Feature Selection into Rule Pruning Methods

In the following we will first show that, four popular used feature selection metrics [39, 8, 15, 27, 31], *DF*, *MI*, *IG*, and χ^2, can be expressed in the terms of support and confidence measures. Based on these expressions, we then present our algorithm to integrate feature selection into the rule pruning phase at the end of this section. Through this method, a feature selection metric can be

derived without the extra computational cost of a separate procedure once the measures of each rule have been computed.

Feature selection techniques can generally be classified into two categories: one selects features within each category, and the other does this among all categories. To deal with the situation of class imbalance, feature selection is performed by selecting the top scoring features based on a specific metric separately for each class in this study, as it was done in [10].

The Transformation of Document Frequency

Document frequency simply measures the number of documents in which the term t occurs [39]. From Definition 1, we immediately have:

$$DF(t, c_i) = lsupp(t, c_i) \tag{1}$$

Lemma 1 *Document frequency of a term t is identical to its local support.*

The Transformation of Mutual Information

The mutual information criterion between the category c_i and the term t is defined as follows [39]:

$$MI(t, c_i) = log\frac{P_r(t \wedge c_i)}{P_r(t)P_r(c_i)} \tag{2}$$

Next, we show that global confidence can serve as a metric identical to mutual information in the same class.

Lemma 2 *Given two association patterns in the same category: r_1: $t_1 \Rightarrow c_i$, and r_2: $t_2 \Rightarrow c_i$, if r_1 has a higher confidence than r_2, then r_1 has a higher mutual information than r_2, vice versa.*

Proof: Let $|c_i|$ be the number of documents with class label c_i, $|T|$ be the total number of all documents, $|t_i|$ be the number of documents with class label c_i that contain term t, and $|t|$ be the total number of all documents that contain term t. Then we have:

$$\begin{aligned} MI(t, c_i) &= log\frac{P_r(t \wedge c_i)}{P_r(t)P_r(c_i)} = log((P_r(c_i)P_r(t|c_i)) - log(P_r(t)) - log(P_r(c_i)) \\ &= logP_r(t|c_i) - logP_r(t) = log(|t_i|/|c_i|)/((|t|/|T|)). \end{aligned}$$

From Definition 6, the mutual information metric can be transformed into the following equation:

$$MI(t, c_i) = log(gconf(t \Rightarrow c_i)/(|c_i|/|T|)). \tag{3}$$

For two rules in the same category, they share the same denominator in equation 3. Since logarithm is a monotonically increasing function, the higher global confidence implies the higher mutual information.

The Transformation of χ^2

χ^2 is a common statistic test that measures the lack of independence between the category c_i and the term t. Given A, the number of times t and c_i co-occur, B, the number of times t occurs without c_i, C, the number of times c_i occurs without t, and D, the number of times neither t nor c_i occurs, the χ^2 statistic is defined to be [39]:

$$CHI(t, c_i) = \frac{|T|(AD - CB)^2}{(A + C)(B + D)(A + B)(C + D)} \tag{4}$$

Lemma 3 *Each component in χ^2 is either a constant, or it is expressed in the terms of support and confidence, so χ^2 can be computed out as an elementary function of support and confidence.*

Proof: According to Definition 1 to 6, we have: $A = |lsupp(t, c_i)|$, $B = |gsupp(t)| - |lsupp(t, c_i)|$, $C = |c_i| - |lsupp(t, c_i)|$, and $D = |T| - |c_i| - B = |T| - |c_i| - |gsupp(t)| - |lsupp(t, c_i)|)$.

The Transformation of Information Gain

Information Gain (IG) measures the number of bits of information obtained for category prediction by knowing the presence or absence of a term t. The information gain of the term t is defined to be [39]:

$$\begin{aligned} IG(t) = &- \sum_{i=1}^{n} P_r(c_i) log P_r(c_i) + P_r(t) \sum_{i=1}^{n} P_r(c_i|t) log P_r(c_i|t) \\ &+ P_r(\bar{t}) \sum_{i=1}^{n} P_r(c_i|\bar{t}) log P_r(c_i|\bar{t}) \end{aligned} \tag{5}$$

For information gain, we have the following lemma.

Lemma 4 *Every component in the information gain formula is either a constant or can be expressed in the terms of support and confidence. In other words, information gain can be computed as a function of support and confidence.*

Proof: The first component in the information gain formula is a constant if the training set is determined. $P_r(t)$, the number of documents containing term t divided by the total number of documents, is identical to the global support. $P_r(c_i|t)$ is the number of documents with class label c_i that also contain term

t, divided by the total number of documents containing t. So we have:

$$P_r(t) = gsupp(t), \quad P_r(\bar{t}) = 1 - gsupp(t)$$

$$P_r(c_i|t) = |lsupp(t, c_i)|/|gsupp(t)| = gconf(t \Rightarrow c_i)$$

$$P_r(c_i|\bar{t}) = |lsupp(\bar{t}, c_i)|/|gsupp(\bar{t})| = (|c_i| - |lsupp(t, c_i)|)/(|T| - |gsupp(t)|)$$

4.4 The Rule Extracting Methods.

After several pruning steps and a sequential covering procedure, we remove redundant and noisy rules. The remaining rules form the end classifiers. The entire procedure of rule extracting, called FARE (Fast Accurate Rule Extracting), is shown in Figure 2.

Fast Accurate Rule Extraction (FARE).

Input: R_i: a set of rules for category i. $minconf$: A confidence threshold. $minphi$: A minimum ϕ coefficient. K: a feature set size.

Output: a subset of rules of the end classifier R_i.

Method:

1. Eliminate rules with confidence less than $minconf$ or ϕ correlation coefficient less than $minphi$ from each R_i.
2. Find the set F_i which includes all items covered by rules with the top K highest value of feature selection metric among all length-1 rules in each R_i
3. Eliminate rules containing items not included in F_i from each R_i.
4. For each rule r in R_i, find its specific rule r' along the vertical directions, if r' has a confidence value lower than r, then remove r' from R_i.
5. Sort the rules in R_i according to their confidence in descending order, while r is not the last rule in R_i, for each rule r ranked first in R_i, if r can correctly classify at least one training instances, then r is selected and the instances covered by r are removed from the training data set T; otherwise r is removed from R_i. If T is empty after removing training instances, then stop the iteration and remove all the rules ranked behind current r from R_i.

Fig. 2. Algorithm for Rule Extraction.

In Figure 2, each rule is evaluated using $gconf$ and phi criteria in line 1. Line 2 computes the value of the metric for each of the features, and features are then ranked in descending order. The K features corresponding to the K best values of metric are then selected, and rules containing items not belonging to the feature set are removed in line 3. Line 4 executes a vertical general-to-specific pruning. At the end of the rule extraction stage, a sequential covering technique is adopted to select the rules in line 5.

Once the classifiers have been constructed, they can be used to predict new documents. The first issue that should be considered in prediction is whether to use multiple rules. In general, a decision made by multiple rules is more reliable. Therefore, in our method, we exploit multiple rules for predicting a test document. However, since we only prune more specific rules with lower confidence during the rule pruning stage, multiple rules having general-to-specific ordering may match the same document. For example, a document in the $'comp.sys.ibm.pc.hardware'$ category of 20NG may be covered by two rules $\{ISA\}$ and $\{ISA, controller\}$ at the same time, and the confidence values for these two rules are 71.58% and 90.91% respectively. In such a case, we should only choose the second rule to score the test document since the second one is more specific and more reliable. In contrast, if no other rules but {ISA} and {bus, SCSI} cover another document in this class, then both rules are combined to label this document since there is no general-to-specific ordering between these two rules.

The second issue is how to score a test document. A simple score model is just to sum matching rules' confidence values, as was done in [2]. However, in case of the skewed class distribution, the number of extracted rules and the distribution of their confidences in different classifiers often vary over a wide range. For a classifier consisting of a few hundred rules whose confidences are distributed from 0.52 to 1, and a classifier consisting of only tens of rules whose confidences distributed from 0.8 to 1, the former is more likely to have a higher recall but a lower precision while the latter is more likely to have a higher precision but a lower recall. We utilize two approaches to solve this problem. The first is to set a bound on confidence. Once the rule with the maximum confidence *maxconf* is found, we set a bound by subtracting from *maxconf* a threshold value τ. Only rules that have confidences higher than this bound can participate in scoring the test document. The second strategy we adopt is to normalize the scores by a normalization factor. The normalization factor (*NormFactor*) for each category, introduced to handle the rule number discrepancy among different classifiers, is defined as the number of rules in this classifier divided by the total number of rules in all classifiers.

The third issue is how to label the test cases without any applicable rule. Our solution to this problem is to merge all antecedents of a classifier into one set which corresponds to a term set of a category called *ClassTermSet*. We then compute the intersection number of *ClassTermSet* and the test document. The class with the maximum intersection number is assigned to the article. The basic idea behind this heuristic is to increase the matching possibility between the test case and the rules. We accomplish this by combining all rules in one classifier into a virtual big rule, and then counting the terms common

Algorithm for Associative Text Categorization

Input: D: A test document data set.
R_i : A set of rule classifiers.
τ : a bound value for maxconf
δ : a parameter for multi-label classification.

Output: Classification Results.

Method:

1. $M_i = \emptyset$, $Score[i] = 0$
2. foreach document d in D
3. foreach rule $r \in R_i$
4. if ($r.antecedent \subseteq d$)
5. $M_i = M_i \cup \{r\}$
6. foreach rule $r \in M_i$
7. foreach rule $r' \in M_i$
8. if ($r.antecedent \subset r'.antecedent$)
9. if ($r.conf < r'.conf$)
10. $M_i = M_i$ - $\{r\}$
11. else
12. $M_i = M_i$ - $\{r'\}$
13. find the rule r_m with $maxconf$ in all M_i
14. foreach rule $r \in M_i$
15. if ($r.conf \geq r_m.conf$ - τ)
16. $Score[i] = Score[i] + r.conf$
17. normalize $Score[i]$ with $NormFactor[i]$
18. find the $maxScore$ in all $Score[i]$
19. if ($Score[i] \geq maxScore$ * δ) assign class label i to d

Fig. 3. Algo. for Associative Text Categorization

to such a big rule and the test document. If the intersection is still null, we then assign the majority class label in the training set or a random one to the test document, depending on whether the training example number is skewed or evenly distributed.

Figure 3 illustrates the algorithm for associative text categorization. In this figure, line 1 initiates some variables, lines 2 - 5 find applicable rules of document d in all classifiers. Lines 6 - 12 find and then select the rule with the highest confidence among all applicable rules that exist with general-to-specific ordering. Line 13 finds the rule with maximum confidence value. Lines 14 - 16 select rules whose confidence is beyond the confidence bound. Lines 17 - 18 normalize the score of selected rules in each classifier and then find the max score value. Line 19 attaches the class label to the new document. For a single-labeled classification task, the threshold δ is just 1 and the class label with the highest score is assigned to this document. For a multi-labeled task, only those with a score exceeding the highest score timing the user-specified value of parameter δ are assigned to the document.

5 Experimental Results

In this section, we present extensive experimental results to show the key design issues related to associative text categorization (ATC) and the performance of our proposed algorithm. Specifically, we demonstrate: (1) the choices of association patterns for ATC, (2) the effect of rule-pruning methods, and (3) a breadth comparison of ATC algorithms.

5.1 Experimental Setup

Experimental Data Sets: We have conducted experiments on three real-world datasets including Reuters-21578, 20NewsGroup (20NG), and WebKB, which are widely used in text categorization research. For the Reuters-21578 data set [20], we select documents from the top-10 largest categories of the Reuters-21578 document collection. We follow the 'ModApte' split which results in a training set of 9603 documents and a test set of 3299 documents. Like other researchers [2, 8, 9, 34], we use the ten largest categories for our study. The 20 Newsgroups (20NG) data set contains approximately 20,000 articles collected from the Usenet newsgroups collection. We use the 'bydate' version [18] of this data set along with its training/test split. The total number of documents in this version is 18846, with a training set size of 11314 and test set of 7532. Finally, the WebKB data set [26] contains web pages, which are divided into seven pre-defined categories. Like many other studies [29, 27, 3], only the four largest categories: student, faculty, course, and project, together containing a total of 4199 pages, are used in our experiment. For each class, we randomly select 80% (3366 web pages) as training data, and we use the remaining 20% (833 web pages) as test data. Both the training and test data consist of four of these universities plus the miscellaneous collection. Such a training/test split is recommended since each university's pages have their idiosyncrasies, and it is easier to train and test on the web pages from the same university than from different universities. For all data sets, we used a stop-list to remove common words [11]. The resulting vocabulary of Reuters-21578, 20-NewsGroup ByDate, and WebKB-4 has 14091, 45120, and 17865 words respectively.

Skewness of Experimental Data Sets The three data sets have their own characteristics, and the corresponding classification tasks differ significantly in their difficulty. Reuters corpus is highly skewed in class sizes. Among the top ten categories, the most common category contains 2877 articles, while the smallest contains only 181 articles. Similar to Reuters-21578, the WebKB-4 data set is also skewed. The largest category contains 1641 web pages, while

the smallest one contains 504 web pages. Messages in the 20NG data set are nearly evenly distributed in all 20 classes. However, different numbers of training examples in different classes are not the only problem responsible for the difficulty of the learning task. The skewed word distribution in one class is also to blame for the loss of performance of learning algorithms. For example, the term sets {div} and {loss, qtr} in the earn class of Reuters, have a local support value of 17.8% and 21.1% respectively and both of them have a global confidence value of 100%. There are many other similar term sets in the Reuters data set. In contrast, although the distribution of training examples in 20NG is nearly even, there are no such words of both high frequency and high differentiate capability in most categories in 20NG data set. Previous work in [19] introduced a new type of class normalization called term-length normalization to solve the imbalance problem. The method proposed there also exploits term distribution among documents in the class, but is designed for a centroid-based classifier.

Table 3

Evaluation Measures

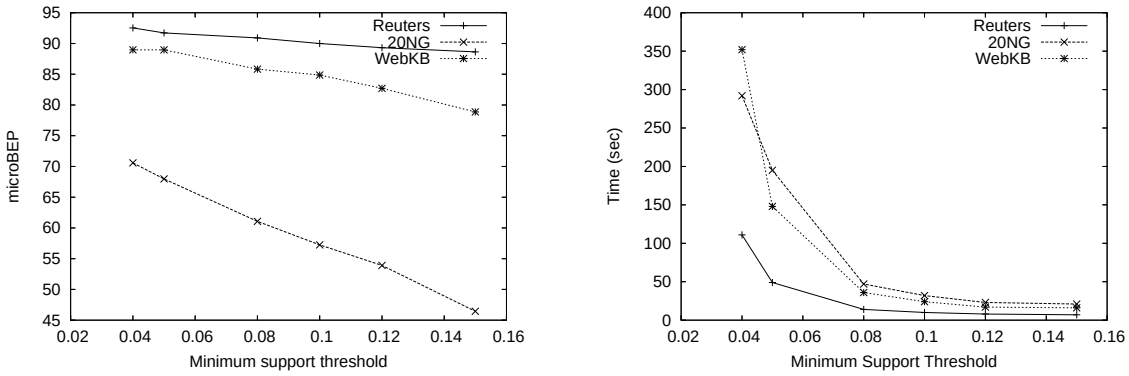
	micro-avg	macro-avg
Precision	$\frac{\sum_{i=1}^{n}(TP_i)}{\sum_{i=1}^{n}(TP_i+FP_i)}$	$\frac{\sum_{i=1}^{n}\frac{TP_i}{TP_i+FP_i}}{n}$
Recall	$\frac{\sum_{i=1}^{n}(TP_i)}{\sum_{i=1}^{n}(TP_i+FN_i)}$	$\frac{\sum_{i=1}^{n}\frac{TP_i}{TP_i+FN_i}}{n}$
BEP	$\frac{microPre+microRec}{2}$	$\frac{\sum_{i=1}^{n}BEP(c_i)}{n}$
F1	$\frac{2*microPre*microRec}{microPre+microRec}$	$\frac{\sum_{i=1}^{n}F_1(c_i)}{n}$

Evaluation Measures: Table 3 shows some commonly used measures, which are based on the concepts of precision (Pre) and recall (Rec). Given a category c_i, let TP_i be the number of items correctly placed in c_i, FP_i be the number of items incorrectly placed in c_i, FN_i be the number of items which belong to c_i but are not placed in c_i, the precision $Pre(c_i) = \frac{TP_i}{TP_i+FP_i}$, and $Rec(c_i) = \frac{TP_i}{TP_i+FN_i}$. F1-measure combines Recall and Precision and is defined as $F_1(c_i) = \frac{2*Pre(c_i)*Rec(c_i)}{Pre(c_i)+Rec(c_i)}$. Finally, the break even point (BEP) [15] is the point where precision equals recall and is computed as reported in [3, 9]. Given a set of Categories $C = \{c_1, c_2, \ldots, c_n\}$, the micro-averaged and macro-averaged [32] formulae for Pre, Rec, BEP, and F1 are listed in Table 3.

Experimental Platform: All experiments were performed on a PC with Pentium IV 1.7GHz CPU and 1.0Gbyte of memory running the Windows 2000 Operating system.

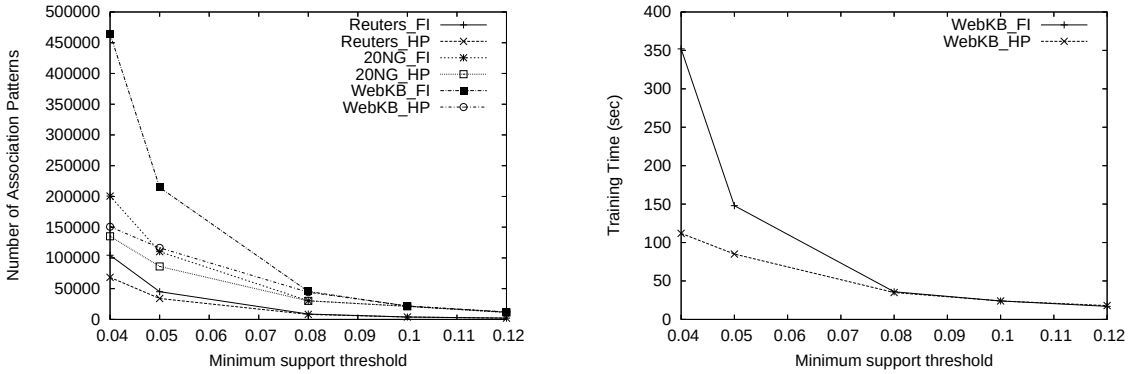
5.2 The Choices of Association Patterns

We first demonstrate the impact of minimum support thresholds on the performance of associative text classification (Abbr. *ATC*) using frequent itemsets (ATC-FIs).



(*a*) (*b*)

Fig. 4. (a) Micro-BEP Values at Different Minimum Support Thresholds for ATC (b) Training Time at Different Minimum Support Thresholds for ATC



(*a*) (*b*)

Fig. 5. (a) A Comparison of Candidate Rule Number (b) A Time Comparison of ATC using FIs and ATC using HPs on WebKB Data Set

Figure 4(a) and Figure 4(b) show microBEP values and training time at different minimum support thresholds on Reuters, 20NG, and WebKB, respectively. In the figure, we can observe: 1) With the decrease of minimum support thresholds, the microBEP values increase steadily for all three data sets. This is due to the fact that low support thresholds can lead to more candidate rules and larger vocabulary coverage; 2) The training time increases very quickly on all data sets as the minimum support thresholds decrease.

Earlier research [14] has revealed that text categorization can benefit from larger vocabulary size. This is also true for associative text categorization as

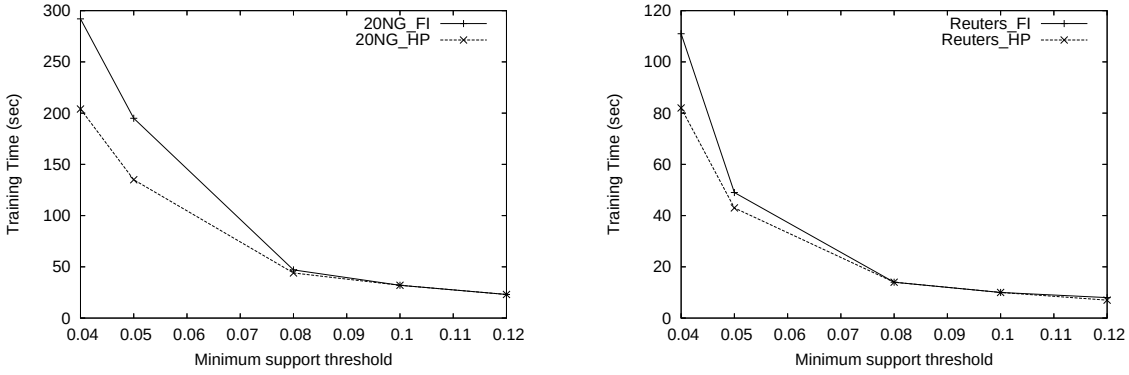
illustrated above. However, the improvement of microBEP is at the cost of losing efficiency. Can we find a way which will keep the competent quality but substantially reduce the amount of candidate rules? With this motivation, we adopt hyperclique patterns (HPs) instead of frequent itemsets (FIs) as candidates in this study. We choose HPs for two reasons: 1) The hyperclique patterns have a strong affinity property [36]. 2) The number of hyperclique patterns is much smaller than that of frequent itemsets, as is demonstrated in Figure 5(a).

The experimental results support our choice. Table 4 shows a BEP comparison of ATC using FIs (ATC-FIs) and ATC using HPs (ATC-HPs). In the table, we can see that both the micro and macro values of ATC-HPs are better than those of ATC-FIs.

Table 4
A Micro-BEP and Macro-BEP Comparison of ATC-HPs and ATC-FIs

	Micro-BEP		Macro-BEP	
Data Sets	ATC-HPs	ATC-FIs	ATC-HPs	ATC-FIs
Reuters	92.57	92.54	87.50	87.07
20NG	70.91	70.58	69.42	69.29
WebKB	89.56	88.96	88.73	87.35

Furthermore, as shown in Figures 6(a),(b) and Figure 5(b), the training time of ATC-HPs is significantly less than that of ATC-FIs on all observed data sets. The computation savings become more dramatic when minimum support thresholds decrease. This indicates that, with the help of hypercliques, we can get better results with a larger vocabulary size by setting *minsup* lower, especially for a sparse dataset such as 20NG.



(a) (b)

Fig. 6. A Time Comparison of ATC using FIs and ATC using HPs (a) 20NG (b) Reuters.

In summary, hyperclique patterns are a better candidate for ATC than frequent itemsets, since it is more efficient to use hyperclique patterns and ATC-

HPs retain the competent classification performance.

5.3 The Effect of Rule-Pruning Methods

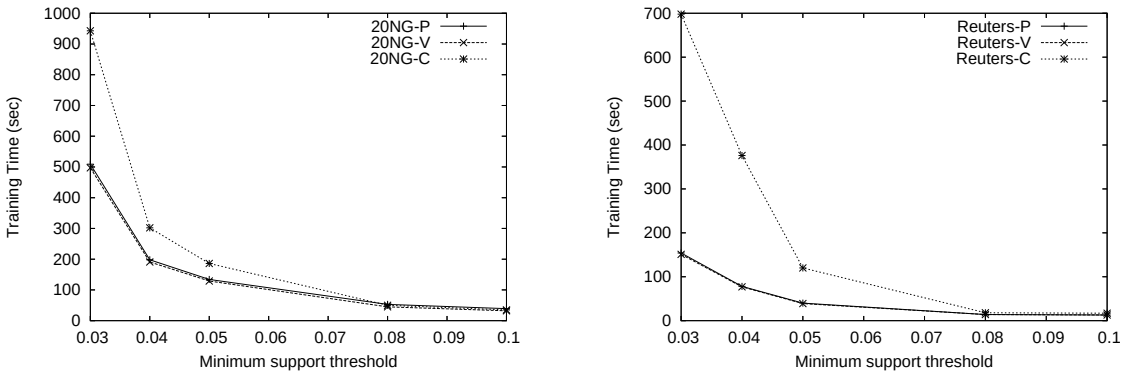
In this subsection, we evaluate the effect of rule-pruning methods on the classification performance. Specifically, there are three designed approaches in this experiment. The first approach, called 'Pure', is a pure version of ATC without applying any rule-pruning strategies. The second is a complete approach, called 'Complete', which integrates both *general-to-specific pruning* and *receiver deciding pruning* into the ATC method. Finally, we also designed a third approach, called 'Vertical', which adopts a *general-to-specific pruning* strategy along vertical paths in the prefix tree when selecting rules and a *receiver deciding pruning* strategy when predicting new instances.

Table 5 shows the comparison of the micro-avg F1 measures obtained using the above three approaches. As can be seen, both $Vertical$ and $Complete$ achieve better micro-avg F1 values than the $Pure$ approach at various different minimum support thresholds. This reveals that general-to-specific pruning can help eliminate noise rules, and thus improve the classification performance of ATC. In addition, we observe that $Vertical$ performs slightly better than $Complete$. Finally, similar results have also been observed on Reuters and 20NG data sets.

Table 5
A Comparison of Pruning Methods on the WebKB Data Set.

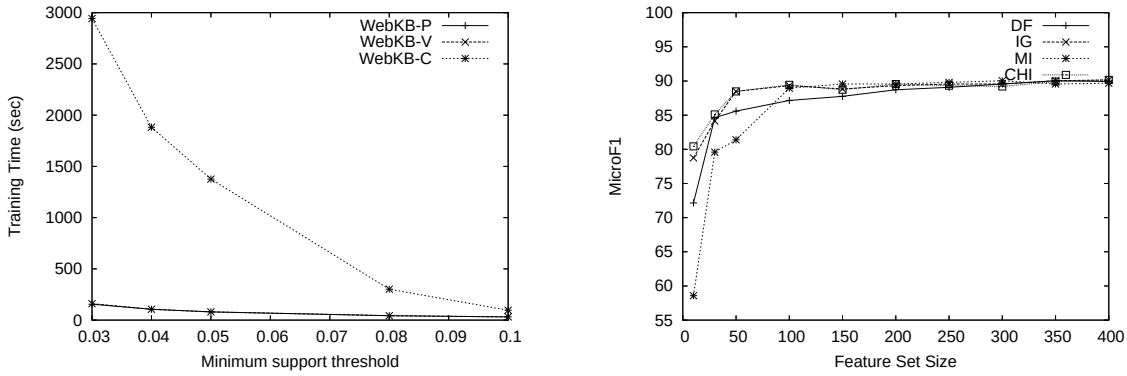
minsup	Pure	Vertical	Complete
0.03	88.84	89.20	88.96
0.04	88.84	90.16	90.16
0.05	88.12	88.96	88.96
0.08	84.51	85.59	85.47
0.10	84.15	84.99	84.99

We also investigated the computational performance of the above three approaches. Figures 7(a),(b) and Figure 8(a) show the training time of three approaches. We can see that the *complete* approach (curve C) is extremely time-consuming compared to the other two approaches. We also observe that the difference between $Pure$ (curve P) and $Vertical$ (curve V) is not significant. For instance, the training time of vertical and pure pruning at *minsup* value 0.03 on WebKB is about $160s$, while it takes near *3000s* for a complete pruning. Hence, $Vertical$ performs the best in terms of the computational performance and classification performance.



(a) (b)

Fig. 7. A Time Comparison of Pruning Methods (a) 20NG (b) Reuters



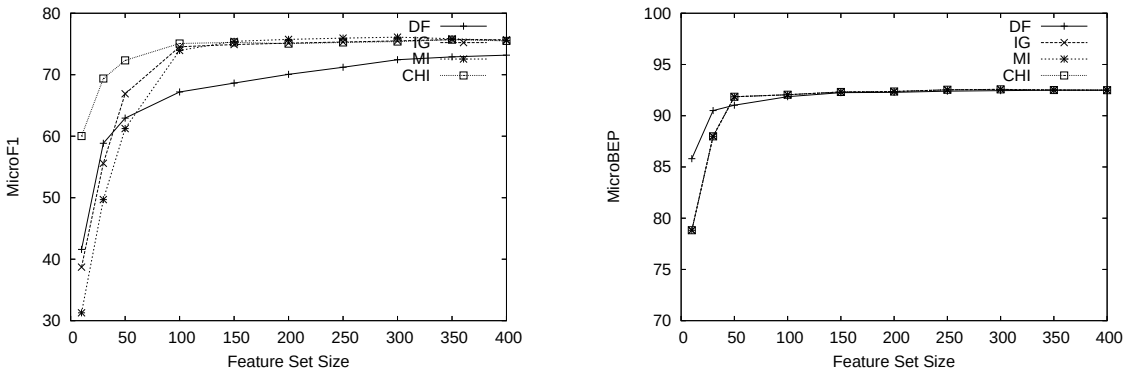
(a) (b)

Fig. 8. (a) A Time Comparison of Pruning Methods on WebKB Data Set (b) The Effect of Feature Selection on WebKB Data Set

5.4 Feature Selection for Associative Text Categorization

We performed experiments with different feature selection methods to examine the proper metric for associative text classification tasks and the vocabulary size related to a certain data set.

Figures 9(a),(b) and Figure 8(b) report the micro-average F1 or BEP of several feature set sizes. From these figures, one can clearly see that, χ^2 criterion performs best among four major metrics on all three data sets, especially at low feature levels. At the outset, this finding seems to differ from the fact that χ^2 is unreliable for rare words. However, this phenomenon can be explained as follows: the candidate rules of an associative classifier come from frequent itemsets, which means that the words with low support (low document frequency) have been eliminated before we select features. The performance of χ^2 here is actually boosted by the combination with DF.



(a) (b)

Fig. 9. The Effect of Feature Selection: (a) 20NG (b) Reuters

We further perform t-test to compare two metrics using the paired F1 values for individual categories of 20NG. Table 6 and Table 7 show the significance tests for small feature set sizes. The results with significant improvements (P value < 0.05) are shown in bold in these two tables. From these tables, we can see that the difference between CHI and other metrics is considered to be extremely statistically significant for a small number of features.

Table 6
Statistic significance in metrics at feature set size 10.

	DF_10	IG_10	MI_10
CHI_10	**0.0001**	**0.0001**	**0.0001**
MI_10	0.6874	**0.0042**	
IG_10	**0.0039**		

Table 7
Statistic significance in metrics at feature set size 30.

	DF_30	IG_30	MI_30
CHI_30	**0.0001**	**0.0001**	**0.0001**
MI_30	0.7155	0.0659	
IG_30	0.1761		

Other noticeable observations from Figure 9 and Figure 8(b) are:

(1) MI has the overall worst performance in most of cases, consistent with prior works in other classification methods [39].
(2) DF is sensitive to the characteristic of the corpus. On one hand, it performs even worse than MI on 20NG, which is a diversified dataset. On the other hand, it yields the best performance on the Reuters data set, for which is well known that very few frequent words are sufficient to yield high accuracy.
(3) On all of the datasets, the graphs for the IG, MI, and CHI metrics exhibit very similar behavior when the feature number is greater than 250.

We found that associative text categorization can also benefit from feature selection. Extensive empirical evaluation on the three data sets shows that the best feature number for Reuters, 20NG, and WebKB at the specific support setting is around 300, 450, and 400, respectively.

5.5 Classifier Comparison

In this subsection, we compare ATC-HPs with some state-of-the-art text categorization algorithms. Table 8 shows the classification results of various classification methods on the Reuters data set. In the table, the overall best result for multi-class categorization of Reuters with a SVM classifier is reported in [8]. The results in Table 8 for Naïve -Bayes, Bayes Net, Decision Tree, and Linear-SVM were obtained from [8]. While the microBEP values of Reuters obtained by other associative text classifiers such as HARMONY [34] and SAT-MOD [9] are also equal to or slightly better than that achieved by SVM, the macroBEP values of these two methods (84.5% and 85.1%) are much less than 87.1% of SVM. In contrast, both the micro-avg (92.6%) and the macro-avg BEP (87.5%) values of our method are better than those of SVM and any other classifiers. Note that micro-avg and macro-avg values emphasize the ability of a classifier on common and rare categories respectively.

Table 8
A Comparison Result on the *Reuters* Data Set (Parameters: minsup=0.04, τ=0.60, δ=0.35, K=280).

category	ATC -HPs	ARC -BC	HAR MONY	SAT -MOD	Naïve Bayes	Bayes Net	Decision Trees	Linear SVM
acq	96.5	90.9	95.3	95.1	87.8	88.3	89.7	93.6
corn	86.2	69.6	78.2	71.2	65.3	76.4	91.8	90.3
crude	87.4	77.9	85.7	90.6	79.5	79.6	85.0	88.9
earn	96.6	92.8	98.1	97.4	95.9	95.8	97.8	98.0
grain	85.8	68.8	91.8	91.3	78.8	81.4	85.0	94.6
interest	78.5	70.5	77.3	74.9	64.9	71.3	67.1	77.7
money-fx	84.4	70.5	80.5	86.6	56.6	58.8	66.2	74.5
ship	87.3	73.6	86.9	83.6	85.4	84.4	74.2	85.6
trade	90.1	68.0	88.4	84.9	63.9	69.0	72.5	75.9
wheat	82.3	84.8	62.8	75.2	69.7	82.7	92.5	91.8
micro-avg	**92.6**	82.1	92.0	92.2	81.5	85.0	88.4	92.0
macro-avg	**87.5**	76.7	**84.5**	**85.1**	74.8	78.8	82.2	**87.1**

We also compare the performance of ATC-HPs and SVM using the WebKB data set. In this experiment, we used C-SVC in LIBSVM [5] as the SVM tool and choose Radial Basis Functions as the kernel function. Table 9 shows the classification performance of ATC-HPs on the WebKB data set. In contrast,

Table 9
The Results of ATC-HPs on the WebKB Data Set (Parameters: minsup=0.04, τ = 0.45, K=400; Training Time: 105 sec; Testing Time: 4 Sec)

category	Precision	Recall	F1
course	90.58	94.02	92.27
faculty	90.65	87.00	88.79
project	92.11	70.00	79.55
student	89.20	96.32	92.63
micro-avg	90.16	90.16	**90.16**
macro-avg	90.64	86.83	**88.31**

Table 10 shows the best results by SVM, which is among the results under the combination of different feature set size and different γ settings. For a fair comparison, we also use information gain as a feature selection metric in the SVM experiment. If we compare the results in the above two tables, we can observe that the macro-F1 (88.39%) of SVM is slightly better than that of ATC-HPs (88.31%). However, the micro-F1 (90.16%) of ATC-HPs is better than that (89.68%) of SVM. Also, SVM needs an additional feature selection procedure, which takes up to 83 seconds.

Table 10
The Results of SVM on the WebKB Data Set (Parameters: γ=0.01, K=450; Feature Selection Time: 83 sec; Training Time: 328 sec; Testing Time: 14 sec).

category	Precision	Recall	F1
course	84.19	88.34	86.21
faculty	81.44	79.00	80.20
project	97.18	93.48	95.29
student	92.00	91.72	91.86
micro-avg	89.68	89.68	**89.68**
macro-avg	88.70	88.13	**88.39**

Table 11 shows the results of ATC-HPs on the 20NG data set. As can be seen, the micro-F1 value of ATC-HPs is 75.77%. Our result is nearly equal to the value (around 76%) in [38], which is achieved by applying SVM and a complex feature selection method called OCFS. Since we do not have the source code from [38], we cannot compare the computation performance. However, due to the nature of SVM, the computation efficiency should be a concern for OCFS.

In summary, ATC-HPs have an equivalent or better performance for text categorization compared to SVM, ATC-FIs, and other classification methods. In addition to this, ATC-HPs have computational advantage over SVM and ATC-FIs. Most importantly, ATC-HPs are rule-based classifiers, and thus are

Table 11
The Results of ATC-HPs on the 20NG-Bydate collection (Parameters: minsup=0.02, $\tau = 0.15$, K=450).

class	Precision	Recall	F1
alt.atheism	61.56	66.77	64.06
comp.graphics	47.48	65.30	54.98
comp.os.ms-windows.misc	69.16	77.61	73.14
comp.sys.ibm.pc.hardware	59.71	62.76	61.19
comp.sys.mac.hardware	81.15	77.14	79.09
comp.windows.x	73.45	72.15	72.80
misc.forsale	80.76	81.79	81.27
rec.autos	77.50	78.28	77.89
rec.motorcycles	84.83	92.71	88.60
rec.sport.baseball	88.81	91.94	90.35
rec.sport.hockey	93.25	93.48	93.37
sci.crypt	82.20	88.64	85.30
sci.electronics	70.93	40.97	51.94
sci.med	84.32	78.79	81.46
sci.space	85.10	85.53	85.32
soc.religion.christian	75.43	77.89	76.64
talk.politics.guns	69.59	84.89	76.49
talk.politics.mideast	91.52	80.32	85.55
talk.politics.misc	77.45	50.97	61.48
talk.religion.misc	67.36	51.79	58.56
micro-avg	75.77	75.77	**75.77**
macro-avg	76.08	74.99	**74.97**

semantic-aware classifiers. The categorization results from ATC-HPs are much more easy to understand and interpret than those of SVM. Meaningful description of the classifiers is clearly desirable for end users. Over the past decades, a great deal of attention has been paid to developing semantic preserving approaches. By using a singular value decomposition of the term-by-document matrix, Latent Semantic Indexing (LSI) [6] can provide information beyond the lexical level and handle effectively with the synonymy problem. Latent Dirichlet Allocation (LDA) [4] uses the Dirichlet distribution to model the distribution of topics for each document. The words in the documents are generated by selecting a topic from this distribution and then choosing a word from that topic. Through this process, LDA can reduce documents to a fixed set of real-valued features and provide topic-based representation of documents. Though these methods have been proved to be very effective for dimension reduction, they need to be combined with some methods such as SVM or kNN for a classification task, That is to say, the classification procedure is still out of the user's control. In contrast, an associative classifier provides users with the opportunity to view or adjust the process of classifi-

cation.

Table 12
Sample Classification Rules

Classification Rules	Local Support	Global Confidence
{inc, commission, shares} ⇒ acq	6.1%	97.1%
{merger, shareholders} ⇒ acq	5.1%	96.6%
{maize, cereals } ⇒ corn	4.4%	100.0%
{agriculture, corn, price } ⇒ corn	9.4%	100.0%
{west, barrel, crude } ⇒ crude	5.1%	100.0%

Table 12 shows some classification rules identified from the Reuters collection using our approach. As can be seen, the classification rules in the table are easy to understand and can be used to interpret the text categorization results. For instance, in the table, there is a classification rule {maize, cereals } ⇒ corn with 4.4% local support and 100.0% global confidence.

6 Conclusions

In this paper, we exploit hyperclique patterns for associative text categorization. Specifically, we first demonstrate why the hyperclique pattern is a better candidate for text categorization than frequent itemsets. Along this line, we design a new algorithm for text categorization using hyperclique patterns. This algorithm deploys a vertical rule-pruning method, which can greatly help reduce the computational cost.

Also, we proposed a feature selection method based on several widely used metrics, which can be mathematically represented in the terms of support and confidence. Moreover, we have integrated this feature selection method into the rule pruning step of our proposed algorithm.

Finally, our experiments on several real-world document data sets showed that our algorithm achieves much better computational performance than state-of-the-art approaches while retaining classification accuracy.

7 Acknowledgements

We sincerely thank the anonymous reviewers and editors for their extensive useful comments and suggestions.

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. of the ACM SIGMOD Conference on Management of Data*, pages 207–216, May 1993.

[2] M. Antonie and O. R. Zaiane. Text document categorization by term association. In *Proc. of the 2nd IEEE International Conference on Data Mining*, pages 19–26, December 2002.

[3] R. Bekkerman, R. El-Yaniv, and N. Tishby. Distributional word clusters vs. words for text categorization. *Journal of Machine Learning Research*, 1:1–48, 2002.

[4] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *Journal of machine Learning Research*, 3:993–1022, 2003.

[5] C-C. Chang and C-J. Lin. Libsvm. In *http://www.csie.ntu.edu.tw/ cjlin/libsvm/*.

[6] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[7] M. Deshpande and G. Karypis. Using conjunction of attribute values for classification. In *Proc. of the 11th International Conference on Information and Knowledge Management*, pages 356–364, November 2002.

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

[9] J. Feng, H. Liu, and J. Zou. Moderate itemset fittest for text classification. In *Proc. of the 14th International World Wide Web Conference*, pages 1054–1055, May 2005.

[10] G. Forman. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3:1289–1305, 2003.

[11] C. Fox. A stop list for general text. *ACM SIGIR Forum*, 24:19–21, 1989.

[12] E. Gabrilovich and S. Markovitch. Text categorization with many redundant features: Using aggressive feature selection to make svms competitive with c4.5. In *Proc. of the 21st International Conference on Machine Learning*, pages 321–328, July 2004.

[13] T. Hofmann. Probabilistic latent semantic analysis. In *Proceedings of*

the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 50–57, August 1999.

[14] T. Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In *Proc. of the 14th International Conference on Machine Learning*, pages 143–151, July 1997.

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

[16] I. T. Joliffe. Principal component analysis. *Springer-Verlag*, 1986.

[17] N. Kwak and ChoiC. H. Feature extraction based on ica for binary classification problems. *IEEE Transaction on Knowledge and Data Engineering*, 15(6):1374–1388, 2003.

[18] Ken Lang. The 20 newsgroups data set. In *http://people.csail.mit.edu/jrennie/20Newsgroups/*.

[19] V. Lertnattee and T. Theeramunkong. Class normalization in centroid-based text categorization. *Information Sciences*, 176:1712–1738, 2006.

[20] D. Lewis. Reuters-21578 text collection. In *http://www.daviddlewis.com/resources/testcollections/ reuters21578/*.

[21] W. Li, J. Han, and J. Pei. Cmar: Accurate and efficient classification based on multiple class-association rules. In *Proc. of the 1st IEEE International Conference on Data Mining*, pages 369–376, November 2001.

[22] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proc. of the 4th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 80–86, August 1998.

[23] B. Liu, W. Hsu, and Ma Y. Pruning and summarizing the discovered associations. In *Proc. of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 125–134, August 1999.

[24] B. Liu, Y. Ma, C. Wong, and P. S. Yu. Scoring the data using association rules. *Applied Intelligence*, 18(2):119–135, 2003.

[25] A. M. Martinez and A. C. Kak. Pca versus lda. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:228–233, 2001.

[26] The CMU WebKB Project. The WebKB data set. In *http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-11/www/wwkb/*.

[27] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *AAAI/ICML-98 Workshop on Learning for Text Categorization*, pages 41–48, July 1998.

[28] D. Meretakis, D. Fragoudis, H. Lu, and S. Likothanassis. Scalable association-based text classification. In *Proc. of the 9th ACM International Conference on Information and Knowledge Management*, pages 5–11, November 2000.

[29] K. Nigam, A.K. McCallum, S. Thrun, and T. M. Mitchell. Learning to classify text from labeled and unlabeled documents. In *Proc. of the 15th Conference of the American Association for Artificial Intelligence*, pages 792–799, 1998.

[30] T. Qian, Y. Wang, H. Long, and J. Feng. 2-ps based associative text clas-

sification. In *Proc. of the 7th International Conference on Data Warehousing and Knowledge Discovery*, pages 378–387, August 2005.

[31] M. Rogati and Y. Yang. High-performing feature selection for text classification. In *Proc. of the 11th International Conference on Information and Knowledge Management*, pages 659–661, July 2002.

[32] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[33] V. D. Silva and J. B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In *Advances in Neural Information Processing Systems*, pages 705–712, 2003.

[34] J. Wang and G. Karypis. Harmony: Efficiently mining the best rules for classification. In *Proc. of the 5th SIAM International Conference on Data Mining*, pages 205–216, April 2005.

[35] K. Wang, S. Zhou, and S. C. Liew. Building hierarchical classifiers using class proximity. In *Proc. of the 25th International Conference on Very Large Data Bases*, pages 363–374, September 1999.

[36] H. Xiong, P. Tan, and V. Kumar. Mining strong affinity association patterns in data sets with skewed support distribution. In *Proc. of the 3rd IEEE International Conf. on Data Mining*, pages 387–394, December 2003.

[37] H. Xiong, P. Tan, and V. Kumar. Hyperclique pattern discovery. *Data Mining and Knowledge Discovery Journal*, 13(2):219–242, September 2006.

[38] J. Yan, N. Liu, B. Zhang, S. Yan, and et al. Ocfs: Optimal orthogonal centroid feature selection for text categorization. In *Proc. of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 122–129, August 2005.

[39] Y. Yang and J. O. Pederson. A comparative study on feature selection in text categorization. In *Proc. of the 14th International Conference on Machine Learning*, pages 412–420, July 1997.

[40] L. Yu and H. Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proc. of the 20th International Conference on Machine Learning*, pages 856–863, August 2003.