Classifying Text Documents by Associating Terms with Text Categories*

Osmar R. Zaïane Maria-Luiza Antonie

Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada.
{zaiane, luiza}@cs.ualberta.ca

Abstract

Automatic text categorization has always been an important application and research topic since the inception of digital documents. Today, text categorization is a necessity due to the very large amount of text documents that we have to deal with daily. Many techniques and algorithms for automatic text categorization have been devised and proposed in the literature. However, there is still much room for improving the effectiveness of these classifiers, and new models need to be examined. We propose herein a new approach for automatic text categorization. This paper explores the use of association rule mining in building a text categorization system and proposes a new fast algorithm for building a text classifier. Our approach has the advantage of a very fast training phase, and the rules of the classifier generated are easy to understand and manually tuneable. Our investigation leads to conclude that association rule mining is a good and promising strategy for efficient automatic text categorization.

Keywords: Text Categorization, Classification, Association Rules, Text Mining.

1 Introduction

The sudden expansion of the Web and the use of the Internet has triggered a lot of research fields, text categorization being only one of them. Text categorization is the distinction between predefined classes of text (documents or paragraphs) by identifying discriminating features in text. After identifying these discriminating features, a text categorizer classifies documents into known classes. A document could fall into one class or many. Automatic categorization of text has been a relevant research issue since the inception of digital documents. A long list of successes and setbacks has been reported in the research literature since the beginning of the 1960s. Many statistical-based and machine-learning-based methods have been proposed and extensive research and experiments were done in text categorization, especially by using natural language processing and artificial intelligence methods, with the main application devoted to support information retrieval. While the enthusiasm for classification of text had almost faded out at one point, the rapid growth of the Web has revived the interest in text categorization. The past decade has seen an increasing effort in applying new techniques in discriminating and classifying text documents. A text categorization system can be used to classify e-mail messages (e-mail response systems), incoming memos, to filter documents, to route texts or to classify web pages in a Yahoo-like manner. The increasing number of the online documents has demanded more research in the text categorization field.

Many techniques have been applied in text categorization, such as Bayesian Networks, decision trees, neural networks, support vector machines, k-nearest neighbor approach, etc. A good survey on these methods and their application in text categorization can be found in [19].

Another aspect that motivates our research is the success of association rule mining in the data mining research community. The use of association rules has been exploited for finding new and interesting hidden patterns in large transactional databases, such as in market basket analysis. Association rules are rules that identify associations between items in transactions. In the recent years many algorithms have been proposed for computing association rules efficiently. Among the proposed algorithms the best known are *apriori* [1] and FP-tree growth [6]. The concept of association rules mining has been intensively used in market basket analysis, but has also found applications in a myriad of domains such as the discovery of topological patterns in images [23].

Automated text categorization represents the process of assigning labels (the labels for each category are predefined) to new documents based on the knowledge accumulated in the training process. That is why building a text classifier necessitates a training set. Both the training and the testing sets must have the documents associated with one or multiple categories. Once the classifier is built using the training set, its effectiveness is determined by comparing the class labels found by the classifier with those being already assigned to the testing set.

In this paper we exploit the use of association rules mining in building a text categorization system from a relatively large training set. Our solution to the text categorization problem differs from the others in the literature in that it consists of modeling documents as transactions and discovering associations between the words existing in documents and the labels assigned to them. The performance study shows that our classifier proves to be efficient especially for the non-overlapping categories, both training and testing phases are very fast and it can be updated incrementally. Moreover, the association rules discovered that

*Research is supported in part by NSERC (the Natural Sciences and Engineering Research Council) of Canada and TeleLearning-NCE (Canadian Networks of Centres of Excellence).

define the classifier are very easy to understand and allow easy manual maintenance and tuning.

The remainder of the paper is organized as follows: Section 2 gives a brief overview of related work in automatic text categorization and in association rule mining. In Section 3 we introduce our text categorization approach using association rules mining. An experiment using Reuters-21578 collection is described in Section 4 along with the performance of our system compared to known systems. We summarize our research and discuss some future work directions in Section 5.

2 Related work

There have been many text classifiers proposed using machine learning techniques: neural networks, genetic algorithms and probabilistic models, etc. Although a lot of approaches have been proposed, automated text categorization is still a major area of research primarily because the effectiveness of current automated text classifiers is not faultless and still needs improvement, and the time to train a classifier is still very significant.

In this section we outline some of the known techniques for categorizing text and the benchmarks used for evaluation. Since our algorithm is based on the association rule mining idea, we briefly present the concepts in order to introduce the subsequent section.

2.1 Text categorization

In the past decade, great attention was paid to the text categorization problem. Most of the text classifiers that were developed and proposed are either machine learning-based or statistical-based. Probably the only one among those that proved to be powerful that comes from information retrieval field is the Rocchio classifier. The problem of building a text classifier was addressed in many ways, but only a few of these have proven to be powerful tools. The techniques applied can be divided in some sub-classes: probabilistic, decision trees, decision rules, regression models, neural networks, support vector machines, etc. [19]. Examples in the literature are [7] using Rocchio Algorithm, [13] using C4.5 decision tree induction, [20] using k-nearest neighbor algorithm, [8] using support vector machines, [22] using neural networks, etc.

Generally, text categorization systems use a vector model representation of the documents. The vector that represents the document contains the document terms and also the weights assigned to each term. The computation of the weights is a problem in itself and will not be covered in this paper. Basically, there are different ways to decide how to assign and how to compute the weights. However, in our model we do not assign weights to terms and model the documents in transactions of unordered terms.

Significant attention was devoted by the research community to the probabilistic classifiers. Even the first text classifier presented in the literature by Maron in 1961 was a probabilistic one [19]. The Naïve Bayesian Classifiers represent an important trend in the probabilistic classifiers. An interesting survey about the different variations in Naïve Bayesian Classifiers was presented in [10]. One drawback of these classifiers is their sensitivity to term space reduction. Term selection can lead to significant degradation in the classification.

Probabilistic classifiers seem rather cryptic for some people and it is not easy to interpret or even maintain the numeric models generated by the classifier. A more intuitive approach in building text classifiers is represented by decision trees and decision rule-based induced classifiers. Many researchers have tackled this class of algorithms [14, 12, 3, 4, 2, 15]. The decision tree classifiers are based on tree induction algorithms such as ID-3 and C4.5 which generate inductive non-numeric rules. Decision rule classifiers come from the machine learning community and they are mainly constructed by inductive rule learning methods.

Some researchers attempted to solve the text categorization problem by using regression models. These classifiers belong to the statistical-based class and probably the best known among them is the 'Linear Least Squares Fit' (LLSF), proposed in [21] in which documents are modeled with two vectors, a vector for weighted terms and a vector of weighted categories, and the classification consists of adjusting the category weights.

Neural networks have been widely used in different domains. Also in text categorization different types of neural networks have been applied in the attempt of making a better classifier, starting by the simplest one - the perceptron, from linear to non-linear neural networks [18]. The problem in using neural networks is that the training time is often excessive and the resulting network is difficult to interpret.

Since the idea of support vector machines was introduced in 1995 by Vapnik, text categorization using support vector machines have also been proposed in the literature [8]. Support vector machines proved to build very powerful systems, which is illustrated by the results presented in [8] and in [19].

Another categorizer, Rocchio classifier [7], often used as a reference for comparison, uses Rocchio's formula for relevance feedback in the document vector space model. Since it was proposed in 1994 many variations have been implemented.

Comparing text classifiers is a difficult and often subjective task. Designers tend to show the bright side of the classifier by putting forward the best examples where the method yields the best results. For instance, it is common that unassigned documents are removed from collections, or only frequent categories are kept for training and testing. An objective way to compare classifiers is to use the same document corpus with fixed categories as well as convene on the same training set and testing set. For the sake of comparison with other methods, we have also applied pruning in the data collections as we shall see later. Most of the proposed classifier systems suggested in the literature have performed experiments on the Reuters and/or OHSUMED text collections for benchmarking. For comparison reasons we performed experiments on both text corpora. In order to construct a learning-based classifier, the collection is divided into a training part and a testing part. In the case of the Reuters collection for example, there exist different splits of the data set that were used for benchmarking: Reuters 22173 "ModLewis", Reuters 22173 "ModApte", Reuters 22173 "ModWiener", Reuters 21578 "ModApte", etc. A good comparison done between well-known text classifiers using this data collection can be found in [19].

2.2 Association Rule Mining

Association rule mining is a canonical data mining task aiming at discovering relationships between items in a dataset. Association rules have been extensively studied in the literature. The efficient discovery of such rules has been a major focus in the data mining research community. From the original *apriori* algorithm [1] there was a remarkable number of variants and improvements culminated by the publication the FP-Tree growth algorithm [6]. Most popular algorithms, however, designed for the discovery of all types of association rules, are apriori-based.

Formally, as defined in [1], the problem is stated as follows: Let $\mathcal{I} = \{i_1, i_2, ...i_m\}$ be a set of literals, called items. Let $\mathcal{D}$ be a set of transactions, where each transaction T is a set of items such that $T \subseteq \mathcal{I}$. A unique identifier TID is given to each transaction. A transaction T is said to contain X, a set of items in $\mathcal{I}$, if $X \subseteq T$. An *association rule* is an implication of the form "$X \Rightarrow Y$", where $X \subseteq \mathcal{I}, Y \subseteq \mathcal{I}$, and $X \cap Y = \emptyset$. The rule $X \Rightarrow Y$ has a *support* s in the transaction set $\mathcal{D}$ is $s\%$ of the transactions in $\mathcal{D}$ contain $X \cup Y$. In other words, the support of the rule is the probability that X and Y hold together among all the possible presented cases. It is said that the rule $X \Rightarrow Y$ holds in the transaction set $\mathcal{D}$ with *confidence* c if $c\%$ of transactions in $\mathcal{D}$ that contain X also contain Y. In other words, the confidence of the rule is the conditional probability that the consequent Y is true under the condition of the antecedent X. The problem of discovering all association rules from a set of transactions $\mathcal{D}$ consists of generating the rules that have a *support* and *confidence* greater that given thresholds. These rules are called *strong rules*.

The idea driving the *apriori* algorithm is to scan the dataset in search of the one-itemsets that are frequent (i.e. support higher than threshold), then combine these to form 2-itemsets called candidates. A second scan of the data set allows to check their support and keep only the frequent ones, then combine them to generate candidate 3-itemsets. the process is repeated again and again for k-itemsets keeping in mind that for a k-itemset to be frequent all its k-1 itemsets have to be frequent, hence the *apriori* nomination. The algorithm only discovers all frequent itemsets in the transactional dataset. Association rule mining is normally a two-step process, wherein the first step frequent item-sets are discovered (i.e. itemsets whose support is no less than the minimum support) and in the second step association rules are derived from the frequent itemsets. The problem of association rule mining is to discover all those association rules that satisfy the conditions of support and confidence.

In our algorithm, as we shall see in the next section, we take advantage of the *apriori* algorithm to discover frequent term-sets in documents. Since the training documents are labeled, we can constrain the association rules to contain the class label in the consequent of the rule and terms in the antecedent. In other words, discovering rules such as "$T \Rightarrow C$", where T is a conjunction of terms from a document and C a class label, would simply generate a classifier for text.

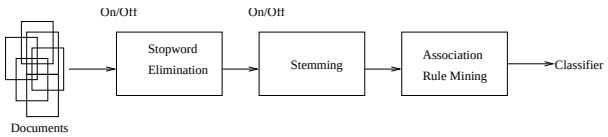

Figure 1: Construction phases for an association-rule-based text categorizer

3 Categorization with Association Rules

3.1 Building the association-rule-based text categorizer

In our approach we construct a text classifier by extracting association rules that associate the terms of a document and its categories. To do so, we model the text documents given for training as a collection of transactions where each transaction represents a text document, and the items in the transaction are the terms selected from the document and the categories the document is assigned to. However, not all terms contained in a document are retained in the transaction. To reduce the term space, we eliminate stopwords and may even reduce some terms to their canonical form by stemming. If a document D_i is assigned to a set of categories $\mathcal{C} = \{c_1, c_2, ...c_m\}$ and after stopwording and stemming the set of terms $\mathcal{T} = \{t_1, t_2, ...t_n\}$ is retained, the following transaction is used to model the document: $D_i : \{c_1, c_2, ...c_m, t_1, t_2, ...t_n\}$.

Initial testing using Porter's stemming algorithm [16] have shown that stemming does not improve effectiveness significantly. We have opted to selectively put stemming, as well as stopwording for that matter, on and off depending upon the data set to categorize. We consider stemming and stopwording as well as the transformation of documents into transactions as a pre-processing phase. The subsequent phase consists of discovering association rules from the set of cleansed transactions. Figure 1 illustrates the phases of the association-rule-based text categorizer construction with the optional stemming and stopwording modules. Stopwords removal considerably reduces the dimensions of the transactional database and thus significantly improves the rule extraction time (i.e. training time). Moreover, while we use a common stopword list in English [5], too frequent terms that are associated to all categories can be automatically added as words to reject. Note that the stopword lists from any language can be used as well.

Using the *apriori* algorithm on our transactions representing the documents would generate a very large number of association rules, most of them irrelevant for classification. We use an apriori-based algorithm that is guided by the constraints on the rules we want to discover. Since we are building a classifier, we are interested in rules that indicate a category label, rules with a consequent being a category label. In other words, given the document model described above, we are interested in rules of the form "$T \Rightarrow c_i$" where $T \subseteq \mathcal{T}$ and $c_i \in \mathcal{C}$. To discover these interesting rules efficiently we push the rule shape constraint in the candidate generation phase of the *apriori* algorithm in order to retain only the suitable candidate itemsets. Moreover, at the phase for rule generation from all the frequent k-itemsets, we use the rule shape constraint again to prune those rules that are of no use in our classification.

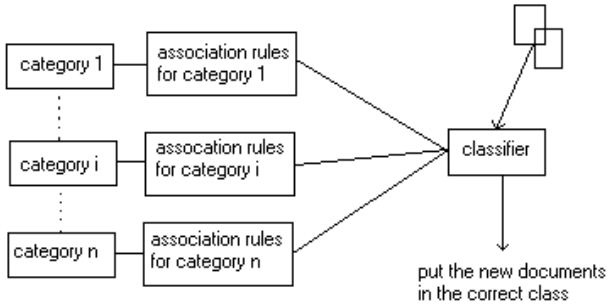

Figure 2: Classifier per category

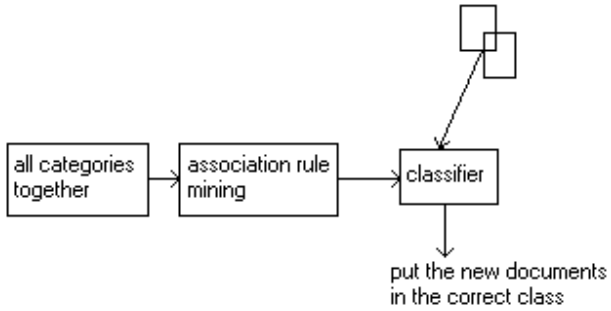

Figure 3: Classifier for all categories

We have considered two different approaches for extracting term-category association rules and for combining those rules to generate a text classifier. In the first approach (Figure 2), each category is considered as a separate text collection and the association rule mining applied to it. In this case, the transactions that model the training documents are simplified to $D_i : \{C, t_1, t_2, ...t_n\}$ where C is the category considered. The rules generated from all the categories separately are combined together to form the classifier. In the second approach (Figure 3) all the categories form a single training collection and the rules generated are *de facto* the classifier. For both approaches we have devised a different algorithm for classification rule discovery. The Association Rule-based Categorizer By Category algorithm (**ARC-BC**), outlined in Algorithm 1, discovers rules for one category at a time and thus does not consider the category items in the document transaction model but just the term items. Once the frequent term-itemsets are discovered, the rules are simply generated by making each frequent term-itemset the antecedent of the rule and the current category the consequent. The Association Rule-based Categorizer for All Categories algorithm (**ARC-AC**), outlined in Algorithm 2 considers all categories combined by using the transactions as described above and pushing the rule form constraint to the candidate generation phase of *apriori*. Basically, any frequent itemset has to include a category label starting from 2-itemsets to k-itemsets. and the join of frequent k-1 itemsets to generate candidate k itemesets is based on the category label in the itemset. While ARC-BC is simple it has to be called for every category in the collection, and it necessitates the split of the training set by category which may add an overhead in the training phase. ARC-AC is obviously faster to train since document transactions are not repeated like with ARC-BC - documents that are assigned to many categories are repeated in ARC-BC. However, ARC-AC has a significant drawback since potentially discriminating terms could be pruned during the frequent itemset generation due to their low support with respect to the whole document collection. Tuning the minimum support threshold could alleviate this problem, but a very low support could generate too many and potentially conflicting rules making the classification slower and less accurate. Moreover the tuning of the minimum support threshold is an iterative and tedious process. The over pruning of relevent terms by ARC-AC explains the reason why ARC-BC slightly outperforms ARC-AC despite the fact that ARC-BC searches for local solutions compared to the global solution of ARC-AC. With both algorithms the document space is reduced in each iteration by eliminating the transaction representing documents that do not contain any of the frequent term-sets. $FilterTable(D_{i-1}, F_{i-1})$ reduces the set of transactions in D_{i-1} that do not contain itemsets in F_{i-1}. This significantly improves the performance of the algorithm (line 6 in Algorithm 1 and line 12 and 16 in Algorithm 2).

Algorithm 3.1 Algorithm 1: (ARC-BC) Find Association rules that associate terms with a given category from a collection of documents all assigned to the same category. **Input:** (i) $\mathcal{D}_1$ a set of transactions representing text documents of the form $\{Cat, t_1..t_m\}$ where Cat is a given category assigned to all the documents and t_i a selected term from the document represented by the transaction; (ii) the minimum support thresholds σ.

Output: Set of rules R such that each rule $r \in R$ is of the form $t_\alpha \wedge t_\beta \wedge t_\gamma \wedge ... \wedge t_\lambda \Rightarrow Cat$ where t_i is a term and Cat is the given category.

Method: The algorithm starts by finding the frequent term-sets then converts each frequent term-set into a rule with the category Cat as the consequent. The pseudo-code for generating the classification rules is as follows:

```
begin
(1)  C_1 ← {Candidate 1 term-sets and their support}
(2)  F_1 ← {Frequent 1 term-sets and their support}
(3)  for (i ← 2; F_{i-1} ≠ ∅; i ← i + 1) do{
(4)       C_i ← (F_{i-1} ⋈ F_{i-1})
(5)       C_i ← C_i − {c | (i − 1) item-set of c ∉ F_{i-1}}
(6)       D_i ← FilterTable(D_{i-1}, F_{i-1})
(7)       foreach document d in D_i do {
(8)            foreach c in C_i do {
(9)                 c.support ← c.support + Count(c, d)
(10)           }
(11)      }
(12)      F_i ← {c ∈ C_i | c.support > σ}
(13) }
(14) Sets ← ⋃_i {c ∈ F_i | i > 1}
(15) R= ∅
(16) foreach itemset I in Sets do {
(17)      R ← R + {I ⇒ Cat}
(18) }
end
```

Algorithm 3.2 Algorithm 2: (ARC-AC) Find frequent term-category-sets for enumerating text association rules in a document labeled collection. **Input:** (i) $\mathcal{D}_1$ a set of transactions representing text documents of the form $\{c_1..c_n, t_1..t_m\}$ where c_i is a category assigned to the document represented by the transaction and t_j a selected term from the document; (ii) the minimum support thresholds σ.

Output: Set of rules R such that each rule $r \in R$ is of the form $t_\alpha \wedge t_\beta \wedge t_\gamma \wedge ... \wedge t_\lambda \Rightarrow C_x$ where t_i is a term and C_j is a category.

Method: The algorithm starts by finding the frequent term-category sets then splits the itemsets to form rules with terms in the antecedent and one category in the consequent. The pseudo-code for generating the classification rules is as follows:

```
begin
(1)  C_0 ← {Candidate categories and their support}
(2)  F_0 ← {Frequent caterories and their support}
(3)  C_1 ← {Candidate  1 term-sets and their support}
(4)  F_1 ← {Frequent 1 term-sets and their support}
(5)  C_2 ← F_0 ⋈ F_1
(6)  foreach document d in D_1 do {
(7)       foreach c in C_2 do {
(8)            c.support ← c.support + Count(c, d)
(9)       }
(10)}
(11)F_2 ← {c ∈ C_2 | c.support > σ}
(12)D_2 ← FilterTable(D_1, F_2)
(13)for (i ← 3; F_{i-1} ≠ ∅; i ← i + 1) do{
(14)      C_i ← (F_{i-1} ⋈ F_2) /* ∀c ∈ C_i c has only one category */
(15)      C_i ← C_i − {c | (i − 1) item-set of c ∉ F_{i-1}}
(16)      D_i ← FilterTable(D_{i-1}, F_{i-1})
(17)      foreach document d in D_i do {
(18)           foreach c in C_i do {
(19)                c.support ← c.support + Count(c, d)
(20)           }
(21)      }
(22)      F_i ← {c ∈ C_i | c.support > σ}
(23)}
(24)Sets ← ⋃_i{c ∈ F_i | i > 1}
(25)R= ∅
(26)foreach itemset I in Sets do {
(27)      R ← R + {W ⇒ C | W ∪ C ∈ I∧
          W is a term-set ∧C is a category in C}
(28)}
end
```

For example, Table 1 illustrates a set of rules obtained as a result of association rule mining algorithms.

net ∧ profit ⇒ earn
agriculture ∧ department ∧ grain ⇒ corn
assistance ∧ bank ∧ england ∧ market ∧ money ⇒ interest
acute ∧ coronary ∧ function ∧ left ∧ ventricular ⇒ myocardial-infarction
ambulatory ∧ ischemia ∧ myocardial ⇒ coronary-disease
antiarrhythmic ∧ effects ⇒ arrhythmia

Table 1: Examples of association rules composing the classifier.

The rules that are generated by the system must have a higher confidence than a certain threshold. In our case we set the threshold to 70% because the higher confidence the stronger the rules. As it can be observed, the antecedent of the rule contains k words, where k words represent a frequent k-itemset that was found applying association rule mining. The consequent of the rules are represented by the categories that are associated with the corresponding k-itemsets. Although the rules are similar to those produced using a rule-based induced system, the approach is different. In addition, the number of words belonging to the antecedent could be large, while in some studies with rule-based induced systems, the rules generated have only one or a pair of words as antecedent [2].

3.2 Classifying a new document

The association-rule-based text categorizer is a set of rules that assigns a category to a document if a set of terms occur in the document. It is common that more than one rule would reinforce the assignment of one document to a class label. Should a document belong to more than one category, many rules with different consequents (i.e. category attached) could be found, for instance "$T_1 \Rightarrow c_i$" and "$T_2 \Rightarrow c_j$" where T_1 and T_2 are term-sets from the document in question. A problem arises when we have too many rules assigning too many categories to a document and we want to limit the number of category assignments per document. To solve this problem, we introduce the notion of dominance factor. The dominance factor δ is the proportion of rules of the most dominant category in the applicable rules for a document to classify. Given a document to classify, the terms in the document would yield a list of applicable rules. If the applicable rules are grouped by category in their consequent part and the groups are ordered by number of rules, the ordered groups would indicate the most significant categories that should be attached to the document to be classified. We call this order category dominance, hence the dominance factor δ. The dominance factor allows us to select among the candidate categories only the most significant. When δ is set to a certain percentage, only the categories that have enough applicable rules representing that percentage of the number of rules applicable for the most dominant category are selected. For example, if $\delta = 0\%$ all categories that have applicable rules are selected. If $\delta = 100\%$, only the most dominant category is selected. More than one category could be selected if the top categories have the same number of applicable rules. When $\delta = 50\%$, only the categories that have the number of applicable rules representing at least 50% of the number of rules of the dominant category are selected. The dominance factor allows adjusting the number of category labels we can assign a document.

4 Experiments with well-known Collection

In order to be able to objectively evaluate our algorithm vis-à-vis other approaches, like other researchers in the field of automatic text categorization, we used the Reuters-21578 text collection [17] and OHSUMED collection as benchmarks. These two databases are described below.

Text collections for experiments are usually split into two parts: one part for training or building the classifier and a second part for testing the effectiveness of the system. There exists many splits of the Reuters collection and we chose to use the "ModApte" version. This split leads to a corpus of 12,202 documents consisting of 9,603 training documents and 3,299 testing documents. There are 135 topics to which documents are assigned. However, only 93 of them have more that one document in the training set and 82% of the categories have less than 100 documents [22]. Obviously, the performances in the categories with just a few documents would be very low, especially for those that do not even have a document in the training set. Among the documents there are some that have no topic assigned to them. We chose to ignore such documents since no knowledge can be derived from them. Finally we decided to test our classifiers on the ten most populated cat-

	CAHI	CNAHI
Category assigned by automated system	a	b
Category not assigned by automated system	c	d

Table 2: Contingency table where CAHI stands for Category assigned by human indexer and CNAHI stands for Category not assigned by human indexer.

egories, categories with the largest number of documents assigned to them in the training set (See Table 3 for the categories). Other researchers have used the same strategy [19], which constrained us to do the same for the sake of comparison. By retaining only the ten most populated categories we had 6488 training documents and 2545 testing documents. On these documents we performed stopword elimination but no stemming.

The other collection used in our experiments is the OHSUMED text collection, which was compiled by William Hersh. This collection was developed at the Oregon Health Science University and it is available online at ftp://medir.ohsu.edu/pub/ohsumed. With a corpus of 348,543 records that have MeSH categories assigned, it consists of Medline records form the years 1987 to 1991. MeSH categories represent human-assigned terms to each record from the Medical Subject Headings (MeSH) vocabulary. From the 348,543 records, only 233,445 of them have abstract. We used in our experiments only those 233,445 records that have both title and abstract. We conducted two major experiments on this data collection. First, we used all the 233,445 documents. The training/testing split used was the same as that reported by other researchers in the literature: the documents from the first four years (183,229) were used for training, while those from 1991 (50,216) are considered for testing. For comparison reasons, we focused in this study on the Heart Disease sub-tree of the Cardiovascular Diseases tree as other researchers reported in their work [11, 9, 18]. The Heart Disease sub-tree contains 119 categories, but due to the variance in frequency of these categories, we used only those categories that had at least 75 documents per category. This new pruning led us to the use of 49 categories out of 119.

On these data sets we tested our two classifiers ARC-BC and ARC-AC. Several measurements have been used in previous studies for evaluation. Some measures, as well as those used in our evaluation, can be defined in terms of precision and recall. The formulas for precision and recall are given below:

recall (R) $= a/(a+c)$; precision (P) $= a/(a+b)$. The terms used to express precision and recall are given in the contingency table Table 2.

For evaluating the effectiveness of our system, we used the F1 measure and breakeven points. F1 measure is a particular case of the F_β measure introduced by van Rijsbergen in 1979. This measure is a combination of precision (P) and recall (R) and has the following formula:

$F_\beta = \frac{(\beta^2+1)\times P\times R}{\beta^2\times P+R}$

F1 measure is obtained when β equals 1. The breakeven point is the point at which precision equals recall. Because we work with many categories and because of comparison reasons we'll compute a macroaverage among the categories. We compute F1 measure for each category separately and then compute the mean of these results.

To evaluate the performance of our text categorization algorithms we used the precision, recall and F_1 measure. F_1 measure combines precision and recall as follows: $F1(p,r) = \frac{2*p*r}{(p+r)}$, where p=precision and r=recall [19]. Table 3 shows a comparison between our ARC-BC classifier and other well known methods. The measure used is the precision/recall-breakeven point on the ten most frequent Reuters categories and macroaveraging performance over all categories. The measures for Bayes, Rocchio, C4.5, k-NN and SVM were obtained from [8]. While our method does not outperform most of the other approaches on this particular Reuters collection, the results are nevertheless quite similar. The results were poor for 'corn' and 'wheat' categories. While for both of these categories the precision was very high, in the order of 95%, the recall was particularly low. This is due to a significant overlap between these two categories. We have not reported the results of ARC-AC in the table because the precision was only 76% on average. This is due to the fact that the Reuters collection is not uniformly distributed. The categories that didn't have enough documents assigned to them in the training set were disadvantaged by our algorithm since the terms didn't have enough support to exceed the minimum support threshold of the association rule mining algorithm. We intend to improve this drawback by adding in the association rule mining process of ARC-AC some partial and relative support thresholds that automatically adapt to the size of the categories. This would avoid dropping terms that have low global support but are actually good class discriminators.

To control our classifiers' compliance to assign multiple categories to documents we have used the dominance factor and we have varied this factor from 0% to 100%, 0 meaning total compliance and assigning all applicable categories, and 100 meaning restricted compliance and assigning only the dominant category. The dominance of a category is in the sense of number of applicable rules. Table 4 shows the results in F_1 measure for the 10 most populated categories with dominance factor 0%, 10%, 25%, 50%, 75% and 100%. One can observe that the performance decreases as the dominance factor is reduced. This is due to the fact that for every document in the testing set there are often too many applicable rules found by the classifier. If the dominance factor is too low, all applicable rules are used and the document ends up categorized in too many categories leading to a low recall. Selecting the categories with the most supported rules but setting the dominance factor to a high level guaranties a higher precision and recall. This problem could be lessened by carefully pruning the association using rule interestingness measures. Good interestingness measures in the context of association rules from textual document are yet to be determined.

While the training time, and even often the testing time, can be very significant for many well-known categorizers, it is important to note that the training for our categorizers is very fast. The training time for our algorithms is in the order of 3 to 4 minutes for the whole Reuters collection depending upon the parameters used. For instance, with a support threshold set to 10% the training of ARC-BC takes 215 seconds on average. The experiments were performed on a PentiumIII 800MHz dual processor machine running Linux. Figure 5 shows the training time as function of the minimum support threshold used to prune the term-sets and the testing time in the same coordinates. The smaller the support threshold, the

Categories	ARC with δ = 0%	10%	25%	50%	75%	100%
acq	45.6	74.8	85.2	90.6	90.9	92.2
corn	4.6	11.7	22.7	35.3	47.7	50.3
crude	14.0	23.7	36.0	61.8	77.9	86.6
earn	62.3	68.1	78.0	90.3	94.2	96.1
grain	12.3	36.3	68.8	75.6	11.0	56.8
interest	9.9	19.8	33.1	51.5	65.2	75.7
money − fx	13.7	26.8	40.6	56.0	64.9	79.5
ship	7.0	17.5	38.3	66.1	66.3	70.8
trade	9.3	18.9	31.0	48.2	63.6	76.2
wheat	6.0	17.4	36.7	52.0	25.5	45.2
MacroAvg.	18.47	31.50	47.04	62.74	60.72	72.94

Table 4: F_1 measure on the ten most populated Reuters categories with a variable dominance factor.

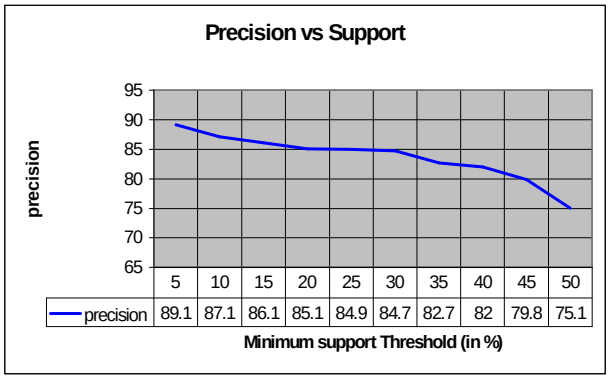

Figure 4: Precision versus the Minimum Support Threshold.

larger the number of frequent itemset discovered and the more time it takes to generate the rules. Figure 4 depicts the variation of the precision the support threshold. We can see that as the support threshold decreases, the precision improves. However, there is a trade-of between precision and training time at the 10% and 5% point. With a support threshold set to 5% the training time reaches 8 minutes on the Reuters collection.

With the OHSUMED collection, our algorithm ARC-BC outperformed the Rocchio classifier (29%) and approximately equaled the neural network classifiers at 40% as reported in the literature by [18]. However, the results were lower than exponentiated gradient (EG) algorithm (50%) and Widrow-Hoff algorithm (55%) also reported by [18].

5 Conclusion and Future Work

This paper introduced the use of association rules in text categorization. Our study provides evidence that association rule mining can be used for the construction of fast and effective classifiers for automatic text categorization. We have presented two algorithms for building the classifier: the first considering categories one at a time, **ARC-BC**, and the second, **ARC-AC**, treats the training set as a whole. Both algorithms assume a transaction-based model for the training document set.

The experimental results show that the association-rule-based classifier performs well and its effectiveness is comparable to most well-known text classifiers. One major advantage of the association-rule-based classifier is its relatively fast training time. Moreover, the rules generated are understandable and can easily be manually updated or adjusted if necessary. The maintenance of the classifier is straight forward. In the case of ARC-BC, when new documents are presented for retraining, only the concerned categories are adjusted and the rules could be incrementally updated.

The experimental results also showed that our association-rule-based text categorizer performed better for those categories that are non-overlapping. The introduction of the dominance factor δ allowed to enhance the for those categories that overlap. Currently, ARC-BC provides a combination of local classifiers and ARC-AC a global classifier that could weed out relevant terms inadvertently. We are in the process of improving ARC-AC by introducing the notion of relative support, a support that adapts to the category at hand rather than a global and universal support threshold. Currently the rules discovered consider the presence of terms in documents to categorize. We are also studying possibilities to take into account the absence of terms in the classification rules as well.

This study demonstrates how promising association-rule-based categorizers could be. We are testing our categorizers on different collections and we intend to study the effectiveness of our algorithms with image collections. Moreover, we plan to enhance our algorithm to handle multi-level association rules in order to categorize text documents with categories defined over a given ontology.

References

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 1994 Int. Conf. Very Large Data Bases*, pages 487–499, Santiago, Chile, September 1994.

[2] C. Apte, F.J. Damerau, and S.M. Weiss. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems*, 12(3):233–251, 1994.

[3] W.W. Cohen and H. Hirsch. Joins that generalize: text classification using whirl. In *4th International Conference on Knowledge Discovery and Data Mining (SigKDD'98)*, pages 169–173, New York City,USA, 1998.

[4] W.W. Cohen and Y. Singer. Context-sensitive learning methods for text categorization. *ACM Transactions on Information Systems*, 17(2):141 –173, 1999.

[5] C. Fox. *Information Retrieval: Data Structures and Algorithms*, chapter Lexical analysis and stoplists, pages 113–116. Prentice-Hall, 1992. ftp://sunsite.dcc.uchile.cl/pub/users/rbaeza/irbook/stopper/.

[6] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *ACM-SIGMOD*, Dallas, 2000.

[7] D. A. Hull. Improving text retrieval for the routing problem using latent semantic indexing. In *17th ACM International Conference on Research and Development in Information Retrieval (SIGIR-94)*, pages 282–289, 1994.

[8] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In *10th European Conference on Machine Learning (ECML-98)*, pages 137–142, 1998.

Categories	*Bayes*	*Rocchio*	*C*4.5	$k - NN$	*SVM*	$ARC - BC$
acq	91.5	92.1	85.3	92.0	95.2	91.3
corn	47.3	62.2	87.7	77.9	85.7	30.5
crude	81.0	81.5	75.5	85.7	88.9	80.7
earn	95.9	96.1	96.1	97.3	98.5	96.4
grain	72.5	79.5	89.1	82.2	92.4	75.8
interest	58.0	72.5	49.1	74.0	76.2	80.5
$money - fx$	62.9	67.6	69.4	78.2	76.2	65.0
ship	78.7	83.1	80.9	79.2	86.5	67.5
trade	50.0	77.4	59.2	77.4	77.1	60.0
wheat	60.6	79.4	85.5	76.6	85.9	25.8
MacroAvg.	65.21	79.14	77.78	82.05	86.26	67.35

Table 3: Precision/recall-breakeven point on the ten most populated Reuters categories for most known classifiers.

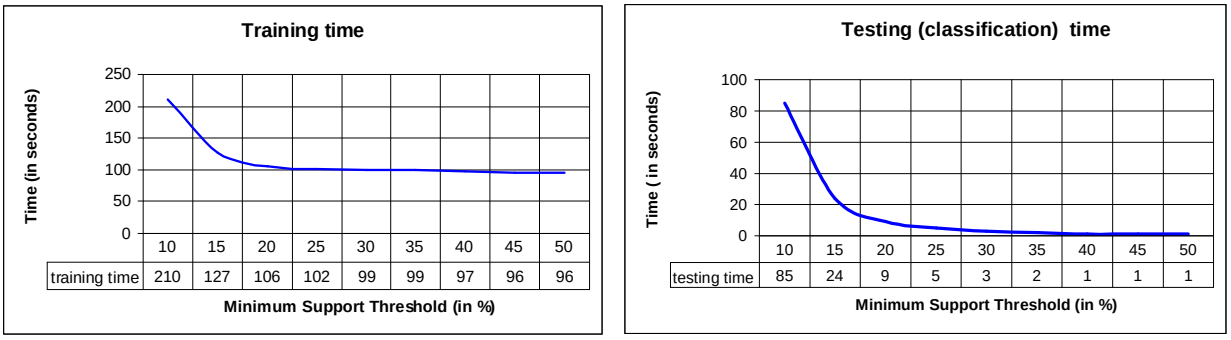

Figure 5: Training and testing time with respect to the support threshold for the whole Reuters-21578 dataset.

[9] W. Lam and C.Y. Ho. Using a generalized instance set for automatic text categorization. In *21st ACM SIGIR International Conference on Information Retrieval*, pages 81–89, Melbourne, Australia, 1998.

[10] D. Lewis. Naïve (bayes) at forty: The independence assumption in information retrieval. In *10th European Conference on Machine Learning (ECML-98)*, pages 4–15, 1998.

[11] D.D. Lewis, R.E. Schapore, J.P. Call, and R. Papka. Training algorithms for linear text classifiers. In *19th ACM SIGIR International Conference on Information Retrieval*, pages 298–306, 1996.

[12] H. Li and K. Yamanishi. Text classification using esc-based stochastic decision lists. In *8th ACM International Conference on Information and Knowledge Management(CIKM-99)*, pages 122 –130, Kansas City,USA, 1999.

[13] T. M. Mitchell. *Machine Learning*. McGraw Hill, New York, US, 1996.

[14] I. Moulinier and J.-G. Ganascia. Applying an existing machine learning algorithm to text categorization. In S.Wermter, E.Riloff, and G.Scheler, editors, *Connectionist statistical,and symbolic approaches to learning for natural language processing*. Springer Verlag, Heidelberg, Germany, 1996. Lecture Notes for Computer Science series, number 1040.

[15] I. Moulinier, G. Raskinis, and J.G. Ganascia. Text categorization: a symbolic approach. In *5th Annual Symposium on Document Analysis and Information Retrieval (SDAIR-96)*, Las Vegas, USA, 1996.

[16] M. F. Porter. Ab algorithm for suffix stripping. *Program*, 14(3):130–137, July 1980. Porter's algorithm can be found at http://www.muscat.co.uk/~martin/stem.html.

[17] The reuters-21578 text categorization test collection. http://www.research.att.com/~lewis/reuters21578.html.

[18] M.E. Ruiz and P. Srinivasan. Neural networks for text categorization. In *22nd ACM SIGIR International Conference on Information Retrieval*, pages 281–282, Berkeley, CA, USA, August 1999.

[19] F. Sebastiani. Machine learning in automated text categorization. Technical Report IEI-B4-31-1999, Consiglio Nazionale delle Ricerche, Pisa, Italy, 1999.

[20] Y. Yang. An evaluation of statistical approaches to text categorization. Technical Report CMU-CS-97-127, Carnegie mellon University, April 1997.

[21] Y. Yang and C.G. Chute. An example-based mapping method for text categorization and retrieval. *ACM Transactions on Information Systems*, 12(3):252–277, 1994.

[22] Y.Yang and X.Liu. A re-examination of text categorization methods. In *22nd ACM International Conference on Research and Development in Information Retrieval (SIGIR-99)*, pages 42–49, Berkeley,US, 1999.

[23] Osmar R. Zaïane, Jiawei Han, and Hua Zhu. Mining recurrent items in multimedia with progressive resolution refinement. In *Int. Conf. on Data Engineering (ICDE'2000)*, pages 461–470, San Diego, CA, February 2000.