A DATA MINING APPROACH FOR CATEGORISING WEB DOCUMENTS

By

KIFAYA SAID QADDOUM

Thesis Supervisor(s):
Dr. Fadi Thabtah
Title: Assistant Professor of Management Information Systems.
Dr. Nadia Yousif
Title: Associate Professor of Computer Science.

In Fulfillment
of the Requirements for the
MASTER Degree in the
Faculty of Information Technology

PHILADELPHIA UNIVERSITY
JAN/2008

A DATA MINING APPROACH FOR CATEGORISING WEB DOCUMENTS

ACKNOWLEDGEMENTS

First of all, I would like to thank Dr. Fadi Fayez, my supervisor and first referee for introducing me to the exciting field of data mining and providing invaluable advice, support and encouragement, as well as for proofreading this thesis drafts. He made this work possible by offering me the opportunity to work on part of his excellent research. I benefitted a lot from the opportunities he provided for me and enjoyed the inspiring working atmosphere he created. I would like to express my gratitude to him for driving me forward in this thesis. Without his help, this work would not have materialized.

I want to extend my warmest thanks to Dr.Nadia Yousif, for reading this thesis and giving insightful comments and suggestions. I am also grateful to her for valuable suggestions and help.

Lastly, to many more people who provided inspiration and emotional support that encouraged me in this data mining journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

KDD	Knowledge Discovery in Databases
AC	Associative Classification
TC	Text Classification
$\|D\|$	Number of Documents
$\|T\|$	Number of terms
tfidf	Term frequency / inverse document frequency
SVM	Support Vector Machines
ti	Term *i*
d_j	Document j
(min sup)	minimum support
TIDS-LIST	transaction id
$\overrightarrow{d_j}$	Support of item (X)
σ(*X*)	Support of item (X)
$D = \{d_1, \ldots, d_r\}$	Set of *r* documents
$C = \{c_1, \ldots, c_q\}$	Set of *q* classes

A DATA MINING APPROACH FOR CATEGORISING WEB DOCUMENTS

ABSTRACT

The core step of KDD is Data Mining. Data Mining applies efficient algorithms to extract interesting patterns and regularities from the data. As volume of information in digital form increases, the use of Text Categorization techniques, which aim at finding relevant information, becomes more necessary. To improve the quality of the classification process form textual data sets, Associative Classification, which utilizes the association rule discovery techniques to construct classification systems, is evaluated in this thesis. Particularly, we developed an associative classification vertical mining algorithm representation in order to improve the accuracy of the classification phase, and to reduce the size of the memory required to store intermediate TIDS in the mining process. Considering the fact that vertical data structure supports fast frequency counting via intersection operations on transaction identifiers (TIDS), this should improve accuracy and decrease memory usage. This thesis demonstrates the problem of using Associative Classification to solve Text Categorization problem, and utilize Diffset structure as a mining approach.

CHAPTER 1

INTRODUCTION

1.1 Motivation

Nowadays, data grow at an alarming speed in various storage devices, and so does valuable information. However, it is difficult to understand hidden information in data without the use of data analysis techniques, which has motivated extensive interest in developing a new field from machine learning. This new field is data mining (DM). Data mining has successfully provided solutions for finding information from text data in bioinformatics, pharmaceuticals, banking, retail, sports and entertainment, etc (Wang et al., 2005) . Many important problems in science and commerce have been addressed by data mining methods, in order to find solutions for the above problems, such methods as Neural Networks (Wiener et al., 1995), Support Vector Machine (SVM) (Joachims, 1998), and Decision Trees (Quinlan, 1993).

In addition to the increasing use of computers, tremendous volumes of data have filled hard disks as digitized information. In the presence of the huge amount of data, the challenge is how to truly understand, integrate, and apply various methods to discover and utilize knowledge from data. To predict future trends and to make better decisions in science, industry, and markets, people are starved for discovery of knowledge from this morass of data. Though 'data mining' is a new term proposed in recent decades, the tasks of data mining, such as classification and clustering, have existed for a much longer time. With the objective to discover unknown patterns from data, methodologies of data mining are derived from machine learning, artificial intelligence, and statistics, etc. The capability of data mining has been proven in improving marketing campaigns, detecting fraud, predicting diseases based on medical records, etc (Wang et al., 2005).

Sales transactions in a retail store are often known as basket data, which can be defined as customer purchases that do not necessarily occur at the same cash point or time (Agrawal, 1993). Consider a retail store with a large collection of sales transactions and customer information. The marketing division at the store is promoting a new credit card in a new geographical area. Typical business decisions have to be made such as how credit card limits are decided for each customer and how each customer's total purchases contribute to the decision process. Classification is a major branch of Data Mining, which used in such cases, finding associations between customer's different features can help the management people in making business decisions. These associations are known as association rules, an example of an association rule is: "55% of customers who buy crisps are likely to buy a soft drink as well; 4% of all database transactions contain crisps and a soft drink". "Customers who buy crisps" is known as rule antecedent, and

"buy a soft drink as well" is known as rule consequent. The antecedent and consequent of an association rule contain at least one item. The 55% of the association rule mentioned above represents the strength of the rule and is known as rule's confidence, whereas the 4% is a statistical significance measure, known as the rule's support.

One subset of the generated Classification Association Rules is chosen to build an automatic model (classifier) that could be used to predict the classes of previously unseen data. This approach, which uses association rule mining to build classifiers, is called associative classification (AC) (Liu, et al., 1998, Li, et al., 2001). Unlike the classic classification approaches such as rule induction and decision trees which usually construct small sized classifiers, AC explores all associations between attribute values and their classes in the training data set, aiming to construct larger sized classifiers, therefore should improve the predictive accuracy within applications (Antonie, et al. 2003; Li, et al., 2001; Yin and Han, 2003). Since (AC), proved to be one of the most efficient techniques in classification; we choose to use it within this work.

As we previously mentioned that the amount of data collected by advanced information systems has increased tremendously. To analyze these huge amounts of data, the interdisciplinary field of Knowledge Discovery in Databases (KDD) has emerged. Thus, new data mining methods are necessary to draw maximum benefit from this additional information. In this chapter, the KDD (Fayyad U., 1996) process is introduced and described. Then data mining and its key tasks are surveyed. A review about Associative Classification (AC) is given. Afterwards the idea of using vertical mining for solving text categorization task is introduced. Finally, the chapter concludes with an outline of the thesis, offering a brief overview of the introduced solutions.

Classification Based on Association (CBA V1.0) is a classification system presented algorithm CBA in the KDD 98, or what is also called AC to deal with text data recently. Text data is converted to database of transactions, and then training and prediction is actually conducted on the derived dataset. In this thesis, the proposed strategy adapts AC to text categorization.

As an example, lets consider the SPAM DETECTION, Spasm or, more formally, unsolicited commercial electronic messages, can undermine the usability of electronic messages. Technical counter-measures include the development of spam filters, which can automatically detect a spam message. Classification algorithms are usually the core of spam filters. The problem is to classify if a given electronic message is spam or legitimate (a message is a data instance). Spasm corresponds to 25% of the total number of messages.

In this thesis, we propose an efficient method for discovering rules based on fast intersection that requires only one database pass and we show by experiments that using Diffset structure (Zaki, M., et al, 2001) reduces physical memory used within transaction storage. Also, using AC with vertical data representation, has showed to be positively effectives on the accuracy of the derived classifiers.

1.2 Knowledge Discovery in Databases

The amount of data collected and stored by electronic devices has risen tremendously during last decades. For example, earth observation satellites retrieving images, bar code scanners collecting costumer data, and companies mapping costumer preferences in data warehouses are generating gigabytes of data every day. Another rapidly growing information collection is the World Wide Web (WWW). Currently the web provides more than 4 billions (Google Press Center) WebPages containing information about almost any imaginable topic.

All of these data collections are far to large to be examined manually and even methods for automatic data analysis based on classical statistics and machine learning often face problems when processing large, dynamic data collections consisting of complex objects. To analyze these large amounts of collected information, the area of Knowledge Discovery in Databases (KDD) provides techniques which extract interesting patterns in a reasonable amount of time. Therefore, KDD employs methods at the cross point of machine learning, statistics and database systems. In (Fayyad U., 1996) KDD is defined as follows:

Knowledge Discovery in Databases *is the non-trivial process of identifying valid, potentially useful, and ultimately understandable patterns in data.*

According to this definition, data is a set of facts that is somehow accessible in electronic form. The term" patterns" indicates models and regularities which can be observed within the data. Patterns have to be valid, i.e. they should be true on new data with some degree of certainty. The potentially usefulness of patterns refers to the possibility that they lead to an action providing a benefit. A pattern is understandable if it is interpretable by a human user. At last KDD is a process indicating that there are several steps that are repeated in several iterations. Figure 1.1(Schubert M., 2004) comprises the following steps:

• Focussing: The first step is to define the goal of the particular KDD task.

• Preprocessing: In this step the specified data has to be integrated, because it is not necessarily accessible on the same system.

• Transformation: The transformation step has to assure that each data object is represented in a common form which is suitable as input in the next step.

• Data Mining: Data mining is the application of efficient algorithms to detect the desired patterns contained within the given data.

• Evaluation: At last, the user evaluates the extracted patterns with respect to the task defined in the focussing step. An important aspect of this evaluation is the representation of the found patterns.

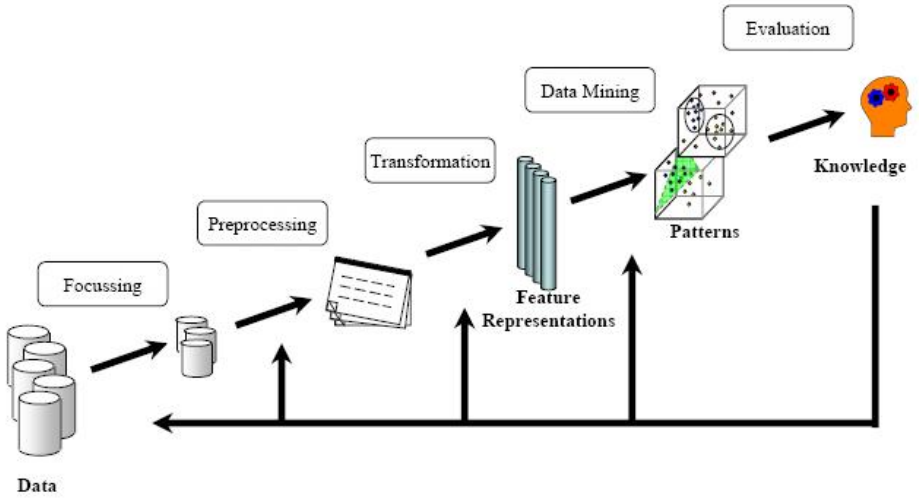


Figure 1.1: The process of KDD.

1.3 Data Mining Overview

1.3.1 Data Mining

Data mining is the most important step within the KDD process, in (Fayyad, U., 1996) Data Mining, and shown in Figure 1.1comes after several steps in KDD, and defined as follows:

Data mining *is a step in the KDD process consisting of applying data analysis and discovery algorithms that, under acceptable computational efficiency limitations, produce a particular enumeration of patterns over the data.*

According to this definition data mining is the step that is responsible for the actual knowledge discovery. To emphasize the necessity that data mining algorithms need to process large amounts of data, the desired patterns have to be found under acceptable computational efficiency limitations. Note that the term data mining and KDD are often used in a synonymous way. In the following, the most important data mining methods are described with respect to the kind of knowledge they mine:

• Classification (also called supervised learning)

Classification is the task of learning a function that maps data objects to one or several classes in a predefined class set (Schubert, M., 2004). To learn this function, classification methods need a training set, containing data objects that are already mapped to the class they belong to. After analyzing the training set, classification methods can map new unknown objects to the classes.

• Clustering (also called unsupervised learning)

Clustering is the task of identifying a finite set of categories (or clusters) to describe the data. Thus, similar objects are assigned to the same category and dissimilar ones to different categories. Clustering is also called unsupervised learning because the data objects are mapped to a set of clusters which can be interpreted as classes as well.

• Association Rules

Finding Association rules is the task of identifying rules that express co-occurrences which means the above-chance frequent occurrence of two terms from a text corpus alongside each other in a certain order, within transaction databases (Schubert, M., 2004). A transaction is a set of items where each item has a different type. Association rules express that in the given database a specified set of items appears together in the same transaction with a certain support/probability. The most important example of transaction data is market basket data.

• Regression

The task of regression is to learn a function which maps data objects to a real value. To find a regression function, a training set of data objects that are already mapped to a real value is necessary (Schubert M., 2004). An additional goal of regression is to discover functional relationships between the feature values of the underlying training objects. Regression is related to classification and Clustering, since these tasks learn functions from a training set.

• Outlier Analysis

Outliers are records in the database that are considerably different from the majority of existing records. Outliers generally have an impact on the calculation of statistics and are discarded in

most cases by the majority of data mining techniques. However, in applications such as fraud detection, the discovery of outliers is important.

KDD comprises more than one phase where data mining is one of its primary phases. Other phases in KDD are data selection, data cleansing, data reduction, pattern evaluation and visualization of the discovered information (Fayyad, et al., 1998; Elmasri and Navathe, 1999). Consider for instance a retail store database where information about customers, including their names, address, postal codes, date of purchase, total paid, etc are stored. During data selection, facts about a group of customers or about customers from specific geographical areas can be selected. Incorrect or incomplete records like invalid postal codes or addresses can be fixed by end users during the data cleansing step. Data reduction decreases the amount of data needed before mining starts, for example, customers may be grouped by income. Once data are processed, then data mining can be applied to derive useful different patterns.

Learning in data mining involves finding and describing patterns from data for different purposes.

The results of the mining phase may be of the following:

- Classification Rules- Customers may be categorized based on payment type; an example of a classification rule is: If income >= 30k and age <= 55 then accept credit card application.
- Association Rules- What items customers are likely to buy together; an example of an association rule is: If a customer buys a pair of jeans and a hat, then he is likely to buy a pair of tennis shoes as well.
- Sequential Pattern- A customer buys a tennis racket, and after one month he buys a digital camera and within four months a tennis shirt. An example of a sequential rule is, “a customer who buys three times in January is likely to buy chocolates on Valentine’s Day”. There is no single data mining technique applicable to all tasks and when it comes to choosing a technique for a particular problem, the choice is very critical as one technique could work well for one problem and poor with others. There are many factors that can be considered before taking such a decision like the size and nature of the data, attribute types (text, real, etc), number of columns, output format and more importantly the goal of application.

1.4 Text Data Mining (TDM)

The appearance and growth of the World Wide Web facilitated the process of spreading and exchanging the information, but apart from that it also gave birth to the completely new ways of communication. These include electronic mail, newsgroups and on-line news, all of which can be stored in text form. The accelerating growth in the amount of text data makes it necessary to automate, at least partially, the process of its search and browsing.

Text expresses a vast, rich range of information, but in its original raw form is difficult to analyze or mine automatically. Most standard DM applications tend to be automated discovery of trends and patterns across large DBs and data sets In the case of text mining, the goal is to look for pattern and trends in large amounts of text (Hearst, 1999) (Wang, J., 2003).

1.4.1 Benefits of TDM

Text mining focuses on how to use a body of textual information as a large knowledge base from which one can extract new, never-before encountered information (Craven, D., et al, 1998). However, the results of certain types of text processing can acquire tools that indirectly aid in the information-access process. As automatically generating term associations to aid in query expansion, and using co-citation analysis to find general topics within a collection or identify central Web pages (Hearst, 1999; Kleinberg, 1998; Larson, 1996).

1.4.2 Methods of TDM

Some of the major methods of TDM include Feature Extraction, Clustering, and Categorization. Feature extraction, which is the mining of text within a document, attempts to find significant and important vocabulary from within a natural language text document (Wang, J., 2003). From the document-level analysis, it is possible to examine collections of documents. The methods used to do this include clustering and classification. In text categorization, the process is a bit more involved. Here, samples of documents fitting into pre-determined "themes" or "categories" are fed into a "trainer," which in turn generates a categorization schema.

1.5 Text Classification

Data mining techniques are used to extract knowledge from relational, otherwise called structured, data where each tuple contains a set of attribute-value pairs. Consider as an example a database with attributes (Sender, Recipient, Date, and Size). The first tuple can be represented as the following set of attribute-value pairs: Sender = “Bob Smith”, Recipient = “Ann Smith”, Date

= “July 12, 2001”, Size = “10”. State of the art decision tree classification algorithm C4.5 (Quinlan, 1993) needs to know attribute of every data element to decide which attribute has the greatest discriminating power. However, different or modified techniques must be applied to text, which is generally not structured according to attribute-value pairs nor has only some structured elements. Examples of unstructured text are an article body, an e-mail message body, and subject fields. Examples of structured elements are an article title, publication date and conference as well as e-mail message sender, recipient and size. Depending on the availability or non-availability of such structured elements, text is also called unstructured or semi-structured data.

Text Mining is concerned with extracting of useful knowledge from structured or semi structured data. TC (Text Classification) is one of the functionalities of Text Mining that can be defined as the supervised learning task of assigning natural language text documents to one or more predefined classes (also called categories or topics) according to their content. The word supervised means that all the data in a training set is preassigned a category before the training process starts (Lewis, D., 1998).

1.6 Associative Classification Mining

The AC approach was introduced to produce rules for describing relationships between attribute values and the class attributes and not for prediction, which is the ultimate goal for classification. In 1998, AC has been successfully employed to build classifiers by (Liu, et al., 1998) and later attracted many researchers, e.g. (Li, et al., 2000; Dong, et al., 1999; Yin and Han, 2003), from data mining and machine learning communities. In this section, we briefly give a formal definition of the AC problem.

1.6.1 Associative Classification Problem

AC is a special case of association rule mining in which only the class attribute is considered in the rule’s consequent (Liu et al., 1998), for example in a rule such as $X \rightarrow Y$, Y must be a class attribute. Let us define the AC problem, where training data set T has m distinct attributes A_1, A_2, A_m and C is a list of class labels. The number of rows in T is denoted $|T|$. Attributes could be categorical (meaning they take a value from a finite set of possible values), such as customer age which could be from (20-50), or continuous (where they are real or integer). In the case of categorical attributes, all possible values are mapped to a set of positive integers. For continuous attributes, a discretisation method is first used to transform these attributes into categorical ones.

Definition 1.1: A row or a training object in T can be described as a combination of

attribute names A_i and values a_{ij}, plus a class denoted by c_j.

Definition 1.2: An item can be described as an attribute name A_i and a value a_i, denoted $< (A_i, a_i)>$.

Definition 1.3: An ITEMSET can be described as a set of disjoint attribute values contained in a training object, denoted $< (A_{i1}, a_{i1}), \ldots, (A_{ik}, a_{ik})>$.

Definition 1.4: A *ruleitem r* is of the form $<ITEMSET, c>$, where $c \varepsilon C$ is the class.

Definition 1.5: The support count (*suppcount*) of *ruleitem r* is the number of rows in *T* that matches *r's* ITEMSETs, and belong to the class *c* of *r*.

Definition 1.6: The occurrence of an ITEMSET *i* (*occitm*) in *T* is the number of rows in *T* that matches *i*.

Definition 1.7: An ITEMSET *i* passes the *minsupp* threshold if $(occitm(i)/|T|) \geq minsupp$.

Definition 1.8: Any *ITEMSET i* that passes the *minsupp* threshold is said to be a *frequent ITEMSET*.

Definition 1.9: Any *ruleitem r* that passes the *minsupp* threshold is said to be a *frequent ruleitem*.

Definition 1.10: A class association rule is represented in the form: $(A_{i1}, a_{i1}) \wedge,\ldots, \wedge (A_{ik}, a_{ik}) \rightarrow c$, where the antecedent of the rule is an *ITEMSET* and the consequent is a class.

Definition 1.11: A training data in classification is considered binary if it contains only two classes (+, -) and each of its training objects is associated with a single class.

Definition 1.12: A training data in classification is considered multi-class if it contains more than two classes and each of its training objects are associated with just one class.

Definition 1.13: Classification accuracy on a test data *Ts*, is the number of cases where the predicted class *p* of each test data *ts* matches *ts* actual class *c* for all cases in the test data.

A classifier is a mapping form $H: A \rightarrow Y$, where *A* is the set of *ITEMSETs* and *Y* is the set of class labels. The main task of AC is to construct a set of rules (model) that is able to predict the classes of previously unseen data, known as the test data set, as accurately as possible.

1.7 Thesis Contributions

There are different issues that arise in AC, including the features of the output, which usually contains only a single class per rule, the efficiency of the algorithms used for this task, the overlapping between rules training objects and the inductive bias of favoring some rules over others in the classifier. These issues are introduced below and are addressed in this thesis.

Classification data is usually highly correlated, and therefore the expected number of potential frequent *ruleitems*, known as candidate *ruleitems*, is relatively large. As a result, it is essential to have a fast algorithm for the discovery of frequent *ruleitems* step in AC. Most of the currently used AC techniques adopt the Apriori (Agrawal and Srikant, 1994) candidate generation step from association rule mining in order to find frequent *ruleitems*. In this thesis we use the terms "frequent *ITEMSET*" and "candidate *ITEMSET*" when talking about association rule algorithms, whereas, we use the terms "frequent *ruleitems*" and "candidate *ruleitems*" when we talk about AC algorithms. In Apriori, the discovery of frequent *ITEMSETs* from transactional databases is accomplished; the aim is to discover associations between items in a transactional database, and to construct a classifier that can forecast the classes of test data objects.

Similarly to association rule mining algorithms, most AC techniques use the Apriori candidate generation step to discover frequent *ruleitems* and due to the repetitive training data search, they suffer from high I/O costs. As a result, some AC techniques employ a more efficient method than Apriori candidate generation step, called FPgrowth (Han, et al., 2000) in order to cut down the number of passes over the training data. In this thesis, we focus on developing an efficient frequent *ruleitems* discovery method, which decreases the number of database scans to one and minimizes the use of complex data structure objects during the learning step, the thesis contributions are summarized below:

- Unlike existing AC techniques that use a horizontal (Agrawal, et al., 1993) format to represent the training data, we use the vertical format representation presented in (Holsheimer, et al., 1995) where each *ITEMSET* has a *TIDS-LIST* transactions that contains that *ITEMSET* in the training data. Empirical studies (Savasere, et al., 1995; Zaki, et al., 1997; Zaki and Gouda, 2003) showed that algorithms that utilize vertical format have shown to be more effective and often better than horizontal techniques, for our thesis we used the vertical format in TC, where it was never used before.

- We propose an efficient fast intersection technique that requires going through the training data only once. Our method stores frequent *ITEMSETs* of size 1 along with their locations (TIDS-LIST) and classes inside arrays during the scan. Then, by intersecting the TIDS-LIST of the frequent *ITEMSETs* of size 1 and using the class labels array, we can easily obtain candidate *ruleitems* of size 2, and so on. A detailed description of the intersection method is given in Chapter 4.

1.8 Document layout

Chapter 2 reviews some existing work related to the thesis. It surveys the fields of Associative Rule Mining and Classification, and major approaches used.

Chapter 3 surveyed Associative Classification approach; this chapter also defines all necessary terminology that is used to describe a classification problem.

Chapter 4 describes the process of VTC classifier construction. First, it describes Data format, *Diffset* technique, and then using vertical mining in text categorization. After that, the chapter introduces the proposed algorithm for classification, which is the core part of VTC classifier. In addition to the last step of the classifier construction, it includes accuracy measurement estimation and the experimental results.

Finally, Chapter 5 concludes with the main observed results and ends with some suggestions for the future research.

CHAPTER 2

LITERATURE REVIEW: ASSOCIATION RULE, AND CLASSIFICATION

2.1 Introduction

Since it has been introduced, Association Rule Mining (ARM) (Agrawal, et al., 1996) has received a great deal of attention by researchers and practitioners among data mining. ARM is an undirected or unsupervised data mining technique, which works on variable length data, and it produces clear and understandable results. It has a simple problem statement, that is, to discover relationships or correlations in a set of items and consequently find the set of all subsets of items or attributes that frequently occur in many database records or examples, and additionally, to extract the rules telling us how a subset of items influences the presence of another subset.

2.2 Association Rule Mining

The association mining task simply can be stated as follows (Agrawal, et al., 1996): Let I be a set of items, and D a database of examples, where each example has a unique identifier (*tid*) and contains a set of items. A set of items is also called an *ITEMSET*. An *ITEMSET* with k items is called a *k-ITEMSET*. The *support* of an *ITEMSET* X, denoted $\sigma(X)$, is the number of examples in D where it occurs as a subset. An *ITEMSET* is *frequent* or *large* if its support is more than a user-specified *minimum support (min sup)* value.

An *association rule* is an expression $A \Rightarrow B$, where A and B are *ITEMSETs*. The support of the rule is the joint probability of an example containing both A and B, and is given as $\sigma (A \cup B)$. The *confidence* of the rule is the conditional probability that an example contains B, given that it contains A, and is given as $\sigma (A \cup B) / \sigma (A)$. A rule is *frequent* if its support is greater than *min sup*, and it is *strong* if its confidence is more than a user-specified *minimum confidence (min conf)*.

2.2.1 Problem Definition

The main objective of data mining is to find interesting/useful knowledge for the user, as Rules are an important form of knowledge; some existing research has produced many algorithms for rule mining. These techniques, use the whole dataset to mine rules and then filter and/or rank the discovered rules in various ways to help the user identify useful ones.

There are many potential application areas for association rule technology which include catalog design, store layout, customer segmentation, telecommunication alarm diagnosis, and so on.

The data mining task is to generate all association rules in the database, which have a support greater than *min sup*, i.e., the rules are frequent, and which also have confidence greater than *min*

conf, i.e., the rules are strong. Here we are interested in rules with a specific item, called the *class*, as a consequent, i.e., we mine rules of the form $A \Rightarrow c_i$ where c_i is a class attribute ($1 \leq i \leq k$).

This task can be broken into two steps:

1. Find all frequent *ITEMSETs* having minimum support for at least one class c_i. The search space for enumeration of all frequent *ITEMSETs* is 2^m, which is exponential in m, the number of items.

2. Generate strong rules having minimum confidence, from the frequent *ITEMSETs*. We generate and test the confidence of all rules of the form $X \Rightarrow c_i$, where X is frequent. For example, consider the sales database of a bookstore (Zaki M., 2000) shown in Figure 2.1, where the objects represent customers and the attributes represent books. The discovered patterns are the set of books most frequently bought together by the customers. An example could be that, "40 percent of the people who buy Jane Austen's *Pride and Prejudice* also *buy Sense and Sensibility*". The store can use this knowledge for promotions, shelf placement, etc.

There are five different items (names of authors the bookstore carries), i.e., $I = \{A, C, D, T, W\}$, and the database consists of six customers who bought books by these authors. Figure 2.1 shows all the frequent *ITEMSETs* that are contained in at least three customer transactions, i.e., *min sup* =50 percent.

DISTINCT DATABASE ITEMS

Jane Austen	Agatha Christie	Sir Arthur Conan Doyle	Mark Twain	P. G. Wodehouse
A	C	D	T	W

DATABASE

Transcation	Items
1	A C T W
2	C D W
3	A C T W
4	A C D W
5	A C D T W
6	C D T

ALL FREQUENT ITEMSETS

MINIMUM SUPPORT = 50%

Support	Itemsets
100% (6)	C
83% (5)	W, CW
67% (4)	A, D, T, AC, AW CD, CT, ACW
50% (3)	AT, DW, TW, ACT, ATW CDW, CTW, ACTW

Figure 2.1: Mining Frequent ITEMSET

FREQUENT ITEMSETS (min_sup = 50%)

Support	Itemsets
100% (6)	C
83% (5)	W, CW
67% (4)	A, D, T, AC, AW CD, CT, ACW
50% (3)	AT, DW, TW, ACT, ATW *CDW*, CTW, *ACTW*

Maximal Frequent Itemsets: ***CDW, ACTW***

ASSOCIATION RULES (min_conf = 100%)

A → C (4/4)	AC → W (4/4)	TW → C (3/3)
A → W (4/4)	AT → C (3/3)	AT → CW (3/3)
A → CW (4/4)	AT → W (3/3)	TW → AC (3/3)
D → C (4/4)	AW → C (4/4)	ACT → W (3/3)
T → C (4/4)	DW → C (3/3)	ATW → C (3/3)
W → C (5/5)	TW → A (3/3)	CTW → A (3/3)

Figure 2.2 Frequent Itemset and Association Rules

Figure 2.2 shows the set of all association rules with *min conf* =100 percent. The *ITEMSETs ACTW* and *CDW* are the maximal frequent *ITEMSETs*. Since all other frequent *ITEMSETs* are subsets of one of these two maximal *ITEMSETs*, we can reduce the frequent *ITEMSET* search problem to the task of enumerating only the maximal frequent *ITEMSETs*.

On the other hand, for generating all the confident rules, we need the support of all frequent *ITEMSETs*. This can be easily accomplished once the maximal elements have been identified by making an additional database pass and gathering the support of all uncounted subsets.

Several algorithms for mining associations have been proposed in the literature. The Apriori algorithm (Agrawal, et al., 1996) is the best known previous algorithm and it uses an efficient candidate generation procedure, such that only the frequent *ITEMSETs* at a level are used to construct candidates at the next level. However, it requires multiple database scans, as many as the longest frequent *ITEMSET*. Some algorithms generate all possible frequent *ITEMSETs*, and for finding the maximal elements use a randomized algorithm to discover maximal frequent *ITEMSETs*.

A number of vertical mining algorithms have been proposed recently for association mining (as well as other mining tasks like classification (Shafer J., et al, 1996). In a vertical database each item is associated with its corresponding *TIDS-LIST*, the set of all transactions (or *TIDS-LIST*) where it appears. Using the vertical format in mining algorithms has shown to be very effective and usually outperform horizontal approaches.

This advantage shows from the fact that frequent patterns can be counted via *TIDS-LIST* intersections, instead of using complex internal data structures (candidate generation and counting happens in a single step). On the other hand, the horizontal approach requires complex search trees. *TIDS-LIST* offer natural pruning of irrelevant transactions as a result of an intersection (*TIDS-LIST* not relevant drop out). Moreover, for databases with long transactions it has been shown using a simple cost model, that the vertical approach reduces the number of I/O operations (Dunkel B., et al, 1999).

Many algorithms use vertical bit-vectors for fast *ITEMSET* and sequence mining respectively. Despite the many advantages of the vertical format, when the *TIDS-LIST* cardinality gets very large (e.g., for very frequent items) the methods start to suffer, since the intersection time starts to become inordinately large. Furthermore, the size of intermediate *TIDS-LIST* generated for frequent patterns can also become very large, requiring data compression and writing of temporary results to disk. Thus (especially) in dense datasets which are characterized by high item frequency and many patterns, the vertical approaches, may quickly lose their advantages.

2.3 Common Association Rule Data Formats

Figure 2.3 illustrates some of the common data formats used in association mining. In the traditional horizontal approach, each transaction has a tid along with the *ITEMSET* comprising the transaction. In contrast, the vertical format maintains for each item its *TIDS-LIST*, a set of all *TIDS-LIST* where it occurs. Most of the past research has utilized the traditional horizontal database format for mining; some of these methods include Apriori (Agrawal, et al., 1996) that mines frequent *ITEMSETs*. A notable exception to this trend is the approaches that use a vertical database format, which include *Eclat* (Zaki M., et al, 2000), Charm, and Partition (Park J., et al, 1995). Our main focus is to improve upon methods that utilize the vertical format for mining frequent patterns.

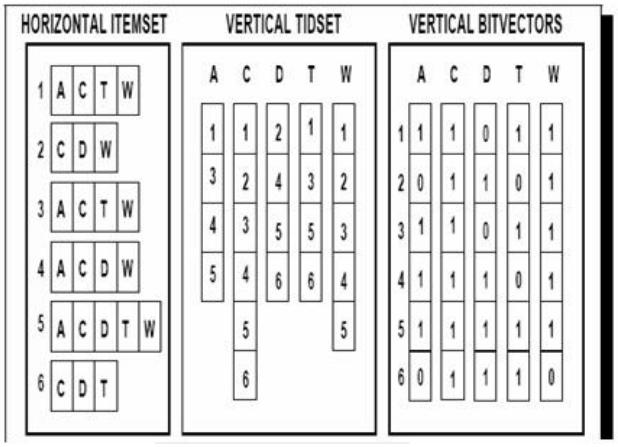


Figure 2.3 Common Data Format

2.4 Existing Algorithms

2.4.1 Apriori

Apriori Algorithm (Liu, B., et al, 2001) is an iterative algorithm that counts *ITEMSETs* of a specific length in a given database pass. The process starts by scanning all transactions in the database and computing the frequent items. Next, a set of potentially frequent candidate 2-*ITEMSETs* is formed from the frequent items. Another database scan is made to obtain their supports. The frequent 2-ITEMSETs are retained for the next pass and the process is repeated until all frequent *ITEMSETs* have been enumerated.

There are three main steps in the algorithm:

1. Generate candidates of length k from the frequent (k-1) length *ITEMSETs*, by a self join on F_{k-1}. For example, If

   F_2= {*AB, AC, AD, AE, BC, BD, BE*}.

   Then

   C3 = {ABC, ABD, ABE, ACD, ACE, ADE, BCD, BCE, BDE}.

2. Prune any candidate with at least one infrequent subset. As an example, ACD will be pruned since CD is not frequent. After pruning, we get a new set

C3 = {ABC, ABD, ABE}.

3. Scan all transactions to obtain candidate supports. The candidates are stored for support counting.

Example Let L3 be {{1 2 3}, {1 2 4}, {1 3 4}, {13 5}, {2 3 4}}. After the join step, C4 will be {{1 2 3 4}, {1 3 4 5}}. The prune step will delete the *ITEMSET* {1 3 4 5} because the *ITEMSET* {1 4 5} is not in L3. We will then be left with only {1 2 3 4} in C4. Figure 2.4 illustrates this example.

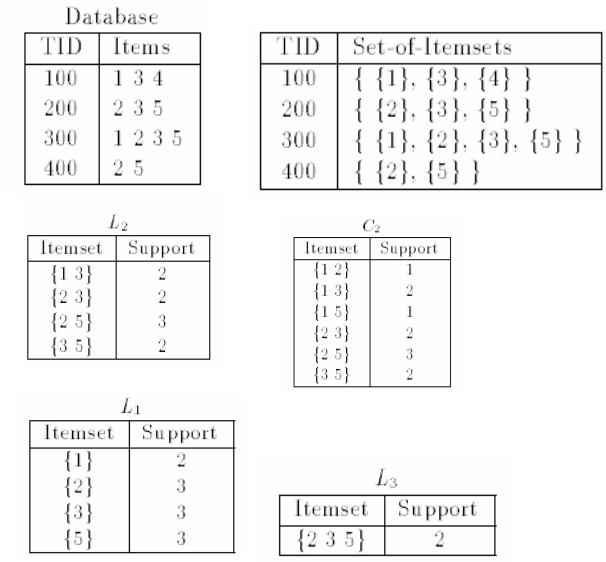


Figure 2.4 Apriori

2.4.2 Frequent Pattern Growth

One of the currently fastest and most popular algorithms for frequent *ITEMSET* mining is the FP-growth algorithm (Han J., 2000) FP-growth algorithm mines frequent patterns without candidate generation and without the high time cost in the generation of candidate *ITEMSETs,* the structure FP-tree, which can facilitate to both the generation of association rules and the update of the mining result by fewer scans of the new data. It is based on a prefix tree representation of the given database of transactions (called an FP-tree), which can save considerable amounts of memory for storing the transactions.

The basic idea of the FP-growth algorithm can be described as a recursive elimination scheme. In a preprocessing step, delete all items from the transactions that are not frequent individually, i.e., do not appear in a user-specified minimum number of transactions. This preprocessing is demonstrated in Table 2.1 and Figure 2.5, which shows an example transaction database. Next step is to select all transactions that contain the least frequent item (least frequent among those that are frequent) and delete this item from them. Recur to process the obtained reduced (also known as projected) database, remembering that the *ITEMSETs* found in the

recursion share the deleted item as a prefix. On return, remove the processed item also from the database of all transactions and start over, i.e., process the second frequent item and so on. In these processing steps the prefix tree, which is enhanced by links between the branches, is exploited to quickly find the transactions containing a given item and also to remove this item from the transactions after it has been processed.

Comparing performance between FP-growth and Apriori on two 10000 record data sets (Han, et al., 2000) indicates that FP-growth is at least an order of magnitude faster than Apriori since the candidate sets that Apriori must maintain become extremely large.

Also the searching process through the database transactions to update candidate *ITEMSETs* support counts at any level becomes very expensive for Apriori, especially when the support

a d f		d a
a c d e	d 8	d c a e
b d	b 7	d b
b c d	c 5	d b c
b c	a 4	b c
a b d	e 3	d b a
b d e	f 2	d b e
b c e g	g 1	b c e
c d f		d c
a b d		d b a

Table 2.1 Transaction database (left), item frequencies (middle), and reduced transaction database with items in transactions sorted descendingly w.r.t. their frequency (right).

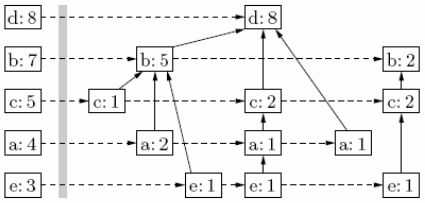


Figure 2.5 FP-Tree for the reduced transaction DB in Table 2.1

threshold is set to a small value. As the number of transactions grows, the processing time difference between the two techniques becomes still larger.

2.4.3 Partition

Partition algorithm (Park J., et al, 1995) is an Apriori-like algorithm that uses TIDS-LIST intersection (Zaki, M., et al, 2001). As described previously, Apriori determines the support values of all candidates of cardinality k-1 before counting the candidates of cardinality k. Instead of counting, Partition uses the TIDS-LIST of the frequent (k-1)-*ITEMSETs* to generate the TIDS-LISTs of the k-candidates by appending single additional item to the frequent (k-1)-*ITEMSETs*. One of the problems with Partition is that when generating TIDS-LISTs of k-candidates, the size of intermediate results easily grows beyond the physical memory limitations of common

machines. Partition overcomes this by splitting the database into several chunks that are treated independently. In the end, an extra scan is required to ensure that locally frequent *ITEMSETs* are also globally frequent.

2.5 Summary

This chapter has presented association rule discovery task in data mining. First part of the chapter has given a general overview on association rule mining and viewed common data representation in transactional data. The second part has surveyed common association rule mining algorithms that employ such approaches in constructing a classifier. Next chapter, will survey different classification approaches that utilize association rule mining to discover the rules. Specifically, we will discuss rule discovery methods, pruning, rule ranking and prediction for numerous AC approaches.

CHAPTER 3

INTEGRATING CLASSIFICATION AND ASSOCIATION RULE MINING

3.1 Introduction

Classification is one of the important tasks of data mining, which has been used for class labels prediction. Classification assigns a new data object to the appropriate class from a dataset which includes labeled data. Whilst in recent years a new classification approach that integrates classification with association rule mining called AC arises, which is considered a promising technique that produces highly accurate classifiers (Baralis, E., et al, 2002).

The rest of the chapter is organized as follows: Classification data mining is surveyed in Section 3.2, main classification techniques is given in Section 3.3, an example to demonstrate the main steps used in AC is illustrated in Section 3.4. Section 3.5 discusses the main challenges and interesting directions in AC, and finally, a chapter summary is given in Section 3.6.

3.2 Classification in Data Mining

Building effective classification systems is one of the main tasks of data mining and machine learning. Past researches have produced many techniques (e.g. Decision Tree (Quinlan. J., 1993), (Yiming, M., 2000), Naive-Bayes (Duda, R., et al, 1973) Support Vector Machines (Cristianini, N., et al, 2000), Neural Networks (Rumelhart, D., et al, 1986) (Lippmann, R., 1987), and Statistical approaches (Quinlan J., 1987),. Classification is one of the analysis methods which use a set of training data and construct a model for each class based on the features in the training data, or it is the process that maps a data item into one of several predetermined classes (Ian H., et al, 2005).

3.2.1 Traditional Problem Decomposition

The classification problem is defined as follows: An input data set called the *training data* consists of a set of multi-attribute records along with a special attribute called the *class* exists. This class attribute draws its value from a discrete set of classes. The training data is used to construct a model (set of rules), which relates the feature variables (or attribute values) in the training data to the class variable. The *test instances* for the classification problem consist of a set of records for which only the feature variables are known while the class value is unknown. The training model is used to predict the class variable for such test instances.

Classification is a well-studied problem see (Ian H., et al, 2005), (Tan P., et al, 2005) for comprehensive overviews and several models have been proposed over the years as stated earlier. Classification task is normally called *supervised* (Ian H., et al, 2005) because, in a sense, the

method operates under supervision by being provided with the actual outcome for each of the training examples—the play or don't play judgment, the lens recommendation, the type of iris, the acceptability of the labor contract. This outcome is called the *class* of the example. The success of classification learning can be judged by trying out the concept description that is learned on an independent set of test data for which the true classifications are known but not made available to the machine. The success rate on test data gives an objective measure of how well the concept has been learned. In many practical data mining applications, success is measured more subjectively in terms of how acceptable the learned description—such as the rules or the decision tree—are to a human user.

Simple Example: The contact lens data shown in Table 3.1 (Ian H., et al, 2005) below tells the kind of contact lens to prescribe, given certain information about a patient. The first column of Table 3.1 gives the age of the patient. In case you're wondering, *presbyopia* is a form of longsightedness that accompanies the onset of middle age. The second column gives the spectacle prescription, *myope* means shortsighted and *hypermetrope* means longsighted. The third column shows whether the patient is astigmatic, and the fourth column relates to the rate of tear production, which is important in this context because tears lubricate contact lenses. The final column (the class attributes) shows which kind of lenses to prescribe: *hard*, *soft*, or none. All

Table 3.1 The contact lens data.

Age	Spectacle prescription	Astigmatism	Tear production rate	Recommended lenses
young	myope	no	reduced	none
young	myope	no	normal	soft
young	myope	yes	reduced	none
young	myope	yes	normal	hard
young	hypermetrope	no	reduced	none
young	hypermetrope	no	normal	soft
young	hypermetrope	yes	reduced	none
young	hypermetrope	yes	normal	hard
pre-presbyopic	myope	no	reduced	none
pre-presbyopic	myope	no	normal	soft
pre-presbyopic	myope	yes	reduced	none
pre-presbyopic	myope	yes	normal	hard
pre-presbyopic	hypermetrope	no	reduced	none
pre-presbyopic	hypermetrope	no	normal	soft
pre-presbyopic	hypermetrope	yes	reduced	none
pre-presbyopic	hypermetrope	yes	normal	none
presbyopic	myope	no	reduced	none
presbyopic	myope	no	normal	none
presbyopic	myope	yes	reduced	none
presbyopic	myope	yes	normal	hard
presbyopic	hypermetrope	no	reduced	none
presbyopic	hypermetrope	no	normal	soft
presbyopic	hypermetrope	yes	reduced	none
presbyopic	hypermetrope	yes	normal	none

possible combinations of the attribute values are represented in the table.

```
If tear production rate = reduced then recommendation = none
If age = young and astigmatic = no and
   tear production rate = normal then recommendation = soft
If age = pre-presbyopic and astigmatic = no and
   tear production rate = normal then recommendation = soft
If age = presbyopic and spectacle prescription = myope and
   astigmatic = no then recommendation = none
If spectacle prescription = hypermetrope and astigmatic = no and
   tear production rate = normal then recommendation = soft
If spectacle prescription = myope and astigmatic = yes and
   tear production rate = normal then recommendation = hard
If age = young and astigmatic = yes and
   tear production rate = normal then recommendation = hard
If age = pre-presbyopic and
   spectacle prescription = hypermetrope and astigmatic = yes
   then recommendation = none
If age = presbyopic and spectacle prescription = hypermetrope
   and astigmatic = yes then recommendation = none
```

Figure 3.1 Rules for the contact lense data.

A sample set of rules learned from this information is shown in Figure 3.1. This is a rather large set of rules, but they do correctly classify all the examples. These rules are complete and deterministic: they give a unique prescription for every conceivable example.

3.3 Classification techniques

3.3.1 Decision trees

A “divide-and-conquer” approach (Quinlan J., 1987) to the problem of learning from a set of independent instances leads naturally to a style of representation called a decision tree (Quinlan, J., 1987). As an example, the decision tree shown in Figure 3.2 has been constructed from the contact lens datasets shown in Table 3.1. Nodes in a decision tree involve testing a particular attribute. Usually, the test at a node compares an attribute value with a constant. However, some trees compare two attributes with each other, or use some function of one or more attributes (Quinlan, J., 1987). Leaf nodes give a classification that applies to all instances that reach the leaf or a set of classifications, or a probability distribution over all possible classifications. To classify an unknown instance, it is routed down the tree according to the values of the attributes tested in

successive nodes, and when a leaf is reached the instance is classified according to the class assigned to the leaf.

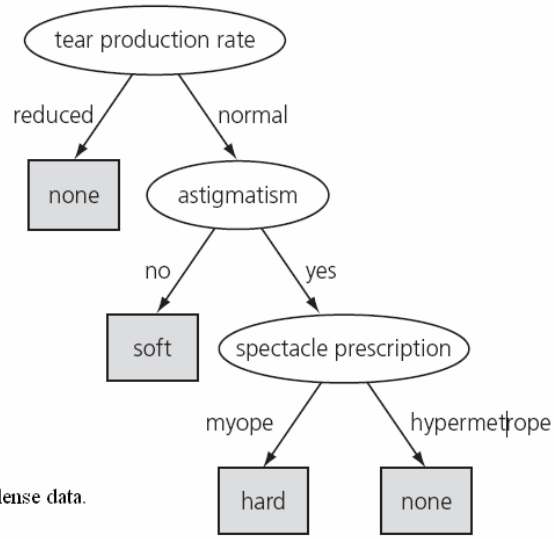


Figure 3.2 Decision Tree for the contact lense data.

Decision trees are particularly suited for data mining. Decision trees can be constructed relatively fast. Another advantage of decision tree models is that they are simple and easy to understand (Quinlan, J., 1993). Decision trees perform a greedy search for rules by heuristically selecting the most promising features. They start with an empty concept description, and gradually add restrictions to it until there is not enough evidence to continue, or perfect discrimination is achieved. Such greedy (local) search may prune important rules.

This direction of classification tries to find a set of rules distinguishing the classes. A decision tree is a tree with the following characteristics:

• Each inner node corresponds to one attribute.

• Each leaf is associated with one of the classes.

• An edge represents a test on the attribute of its father node.

For classification, the attribute values of a new object are tested beginning with the root. At each node the data object can pass only one of the tests that are associated to the departing edges. The tree is traversed along the path of successful tests until a leaf is reached. To construct a decision tree there are multiple approaches like (Gehrke, J., et al, 1998), (Breiman, L., et al, 1984). Most of these approaches split the training set recursively by selecting an attribute. The training set is now split by the values of the selected attribute. To determine the attribute the most promising candidate with respect to a given quality criteria is determined. An example quality criterion is the information gain, which is introduced in section 2.1.2. This split step is done

recursively for all subsets until a breaking criteria is reached or the members of a subset strictly belong to a class. Finally, more sophisticated approaches prune the decision tree to avoid overfitting and find a smaller model. Note that this approach to decision tree construction does not necessarily create the smallest decision tree possible. However, the problem of finding a minimal decision has an exponential time complexity and the introduced heuristic solutions yield good classification accuracy in many cases.

For decision tree technique, it is easy to read a set of rules directly off a decision tree. One rule is generated for each leaf. The antecedent of the rule includes a condition for every node on the path from the root to that leaf, and the consequent of the rule is the class assigned by the leaf. This procedure produces rules that are unambiguous in that the order in which they are executed is irrelevant. However, in general, rules that are read directly off a decision tree are far more complex than necessary, and rules derived from trees are usually pruned to remove redundant tests. Because decision trees cannot easily express the disjunction implied among the different rules in a set, transforming a general set of rules into a tree is not quite so straightforward. A good illustration of this occurs when the rules have the same structure but different attributes, like:

If a and b then x0
If c and d then x

Information Gain

The information gain is measure for the degree a given term is capable to distinguish the classes. It is based on the entropy as a measure of pureness with respect to set of classes C.

Definition 3.1 (entropy)

Let DB be a set of documents and let C = {C1, ..Cm} with DB = $\cup 1 \leq_i \leq_m C_i$ be a disjunctive partitioning of DB, Pr is the probability of occurrence of C. Then the entropy of DB with respect to the partitioning C is defined as follows:

$$entropy(C) = -\sum_{i=1}^{m} \Pr[Ci] \cdot \log PR[Ci] \qquad \textbf{3.1}$$

where $Pr[C_i]$ denotes $\frac{|C_i|}{|DB|}$.

Definition 3.2 (Information Gain)

Let t be a term and let cont(t, S) = {d ∈ S|t ∈ d} denote the subset of set S that contains a term t and let cont(t, S) denote S \ cont(t, S). The information gain of t with respect to the disjunctive partitioning C is:

$$GC(t) = entropy(C) - \frac{|cont(t,DB)|}{|DB|} \cdot entropy(cont(t,DB)) - \frac{|\overline{cont(t,DB)}|}{|DB|} \cdot entropy(\overline{cont(t,DB)}) \quad \textbf{3.2}$$

The idea of information gain is to split a given set according to the occurrence of a given term and afterwards compare the weighted average of the resulting subsets to the entropy of the unsplitted set. If the entropy in the subsets decreases significantly, the term provides a higher information gain and is better suited as a feature.

Figure 3.3 (Ian, H., et al, 2005) shows individual rules as being effectively logically ORed together: if any one applies the condition, the class (or probability distribution) given in its conclusion is applied to the instance. However, conflicts arise when several rules with different conclusions are applicable to a test case.

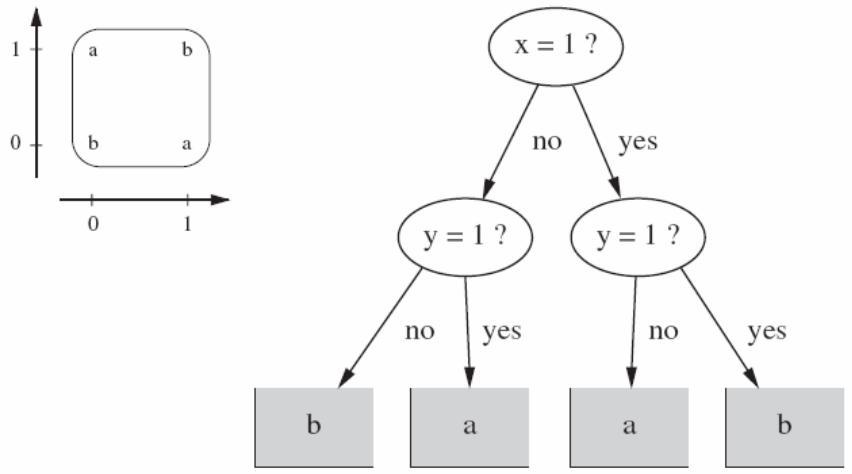


Figure 3.3 The exclusive-or problem

The advantages of decision trees are that they are very robust against attributes that are not correlated to the classes because those attributes will not be selected for a split. Another more important feature is the induction of rules. Each path from the root to a leaf provides a rule that can be easily interpreted by a human user. Thus, decision trees are often used to explain the characteristics of classes.

The drawback of decision trees is that they are usually not capable to consider complex correlations between attributes because they only consider one attribute at a time. Thus, decision trees often model correlated data by complex rules which tend to overfitting. An more detailed discussion of decision trees is found in (Han, J., et al, 2001).

3.3.1.1 C4.5 Algorithm

C4.5 is an algorithm used to generate a decision tree developed by (Quinlan. J., 1993). C4.5 is an extension of Quinlan's earlier ID3 algorithm (Quinlan, J., 1986).

C4.5 algorithm is considered on of the best decision tree methods for extracting rules from a data set. Modification to C4.5 named "C5" has been developed by (Quinlan. J., 1993). As for the ID3 algorithm, C4.5 uses information gain to select the root attribute.

C4.5 builds decision trees from a set of training data in the same way as ID3, using the concept of Information Entropy. The training data is a set $S = s_1,s_2,...$ of already classified samples. Each sample $s_i = x_1,x_2,...$ is a vector where $x_1,x_2,...$ represent attributes or features of the sample. The training data is augmented with a vector $C = c_1,c_2,...$ where $c_1,c_2,...$ represent the class that each sample belongs to.

C4.5 uses the fact that each attribute of the data can be used to make a decision that splits the data into smaller subsets. C4.5 examines the normalized Information Gain (difference in entropy) that results from choosing an attribute for splitting the data. The attribute with the highest normalized information gain is the one used to make the decision. The algorithm then recurses on the smaller sub lists.

This algorithm has a few base cases; the most common base case is when all the samples in a given list belong to the same class. Once this happens, the solution simply is to create a leaf node

In pseudo code the algorithm looks like this:

```
Check for base cases
For each attribute a
        Find the normalized information gain from splitting on a
Let a_best be the attribute with the highest normalized information gain
Create a decision node node that splits on a_best
recurse on the sublists obtained by splitting on a_best and add those nodes as children of node
```

for the decision tree telling to choose that class. It might also happen that none of the features

give you any information gain; in this case C4.5 creates a decision node higher up the tree using the expected value of the class. It also might happen that you've never seen any instances of a class; again, C4.5 creates a decision node higher up the tree using expected value.

Information Gain and Information Entropy

As explained earlier in previous sections, *Entropy(S)* can be thought of as a measure of how random the class distribution is in *S*. Information gain is a measure given to an attribute *a*. *a* Can separate *S* into subsets S_a1,S_a2,S_a3,...,s_an the information gain of a is then *Entropy(S)* - *Entropy*(S_a1) - *Entropy*(S_a2) - ... - *Entropy*(S_an). Information gain is then normalized by multiplying the entropy of each attribute choice by the proportion of attribute values that have that choice. Missing values are treated by C4.5 using probabilities that are computed based on the frequencies of the different values for an attribute at a particular node in the decision tree (Witten, I., et al, 2000). Consider an attribute P that has a missing data object P (p) in the training data. C4.5 algorithm assigns a probability to each values of P using the observed frequencies at a node n. For instance, suppose attribute P has three possible values at node n and n contains ten known examples with five P = 20, two with P = 30 and three with P =50. C4.5 then considers the probability that P(p)=20 is 0.5, the probability that P(p)=30 is 0.2 and the probability that P(p)=50 is 0.3. Each of these fractions is distributed to each possible branch for attribute P down in the tree. Finally, C4.5 algorithm uses these fractional examples for the process of estimating information gain.

3.3.1.2 C4.5 Improvements over the ID3 Algorithm

C4.5 made a number of improvements to ID3. Some of these are:

- Handling both continuous and discrete attributes - In order to handle continuous attributes, C4.5 creates a threshold and then splits the list into those whose attribute value is above the threshold and those that are less than or equal to it. (Quinlan. J., 1993).
- Handling training data with missing attribute values - C4.5 allows attribute values to be marked as? For missing. Missing attribute values are simply not used in gain and entropy calculations.
- Handling attributes with differing costs.

- Pruning trees after creation - C4.5 goes back through the tree once it's been created and attempts to remove branches that do not help by replacing them with leaf nodes.

Continuous attributes are discretized using a discretization method such as (Fayyad, U., et al, 1993). One of the major extensions of the ID3 algorithm that C4.5 proposed is that of pruning. Two known pruning methods used by C4.5 to simplify the decision trees constructed are sub-tree replacement and pessimistic error estimation (Quinlan. J., 1993).

3.3.2 Classification rules

Rule classification (Huber, K., et al, 1995) seeks to present data in such a way that interpretations are actionable and decisions can be made based on the knowledge gained from the data. For data mining clients, they expect a simple explanation of why there are certain

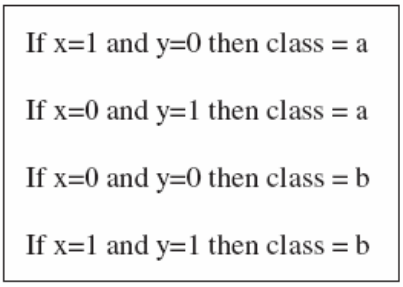


Figure 3.3.1 Classification Rules

classification results: what is going on in a high-dimensional database, and which feature affects data mining results significantly, etc. For example, a succinct description of a market behavior is useful for making decisions in investment. A classifier learns from training data and stores the learned knowledge into the classifier parameters, such as the weights of a neural network classifier. However, it is difficult to interpret the knowledge in an understandable format by the classifier parameters. Hence, it is desirable to extract IF–THEN rules to represent valuable information in data as Figure 3.3.1 shows.

Definition 3.3 (*Classification Context*). Let *T, I*, and *C* be respectively a finite set of transaction identifiers (TIDS-LIST) *t*, of items *i*, and class labels *c*. The classification rule context can be formalized as $D = (T, I, C, L_i, L_c)$. Each couple (t, i) in L_i, $L_i \subseteq T \times I$, expresses the occurrence of item *i* in transaction *t*. Each couple (t, c) in L_c, $L_c \subseteq T \times C$, expresses the occurrence of class label *c* in transaction *t*. A transaction can contain a single class label.

The *antecedent,* or precondition, of a rule is a series of tests just like the tests at nodes in decision trees, and the *consequent,* or conclusion, gives the class or classes that apply to instances

covered by that rule, or perhaps gives a probability distribution over the classes. Generally, the preconditions are logically ANDed together, and all the tests must succeed if the rule is to be applied. However, in some rule formulations the preconditions are general logical expressions rather than simple conjunctions.

3.3.3 Common Classification Rule Approaches

3.3.3.1 Repeated Incremental Pruning to Produce Error Reduction.

RIPPER (Repeated Incremental Pruning to Produce Error Reduction), was proposed by (Cohen. W., 1995).It consists of two main stages: the first stage constructs an initial ruleset using a rule induction algorithm called IREP* (Cristianini, N., et al, 2000); the second stage further optimizes the ruleset initially obtained. These stages are repeated for k times. IREP* is called inside RIPPER-k for k times, and at each iteration, the current dataset is randomly partitioned in two subsets: a growing set, that usually consists of 2/3 of the examples and a pruning set, consisting in the remaining 1/3. These subsets are used for two different purposes: the growing set is used for the initial rule construction (the rule growth phase) and the pruning set is used for the pruning (the rule pruning phase). IREP* uses MDL (Clark, P., et al, 1993) as a criterion for stopping the process.

The rule growth phase: The initial form of a rule is just a head (the class value) and an empty antecedent. At each step, the best condition based on its information gain is added to the antecedent. The stopping criterion for adding conditions is either obtaining an empty set of positive instances that are not covered or not being able to improve the information gain score. The rule pruning phase: Pruning is an attempt to prevent the rules from being too specific. Pruning is done accordingly to a scoring metric denoted by v*. IREP* chooses the candidate literals for pruning based on a score which is applied to all the prefixes of the antecedent of the rule on the pruning data.

3.3.4 Statistical Approach

3.3.4.1 Naïve Bayes

Naïve Bayes (Duda, R., et al, 1973), is considered as a statistical classification approach, that has been given this name because it's based on Bayes's rule and "naïvely" assumes independence—it is only valid to multiply probabilities when the events are independent.

The assumption that attributes are independent (given the class) in real life certainly is a simplistic one. But despite the disparaging name, (Meretakis D., et al, 1999) works very well when tested on actual datasets, particularly when combined with some of the attribute selection procedures that eliminate redundancy, and hence nonindependent, attributes. One thing that can go wrong with Naïve Bayes is that if a particular attribute value does not occur in the training set in conjunction with every class value, things go badly awry. Table 3.2 shows a simple data set about the weather problem (Ian H., et al, 2005).

Table 3.2 The weather data.

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Table 3.3 (Ian H., et al, 2005) shows a summary of the weather data obtained by counting how many times each attribute–value pair occurs with each value (yes and no) for

Table 3.3 The weather data with counts and probabilities.

Outlook			Temperature			Humidity			Windy			Play	
	yes	*no*		*yes*	*no*		*yes*	*no*		*yes*	*no*	*yes*	*no*
sunny	2	3	hot	2	2	high	3	4	false	6	2	9	5
overcast	4	0	mild	4	2	normal	6	1	true	3	3		
rainy	3	2	cool	3	1								
sunny	2/9	3/5	hot	2/9	2/5	high	3/9	4/5	false	6/9	2/5	9/14	5/14
overcast	4/9	0/5	mild	4/9	2/5	normal	6/9	1/5	true	3/9	3/5		
rainy	3/9	2/5	cool	3/9	1/5								

play. For example, it is obvious from Table 3.2 that outlook is sunny for five examples, two of which have play = yes and three of which have play = no. The cells in the first row of the new table simply count these occurrences for all possible values of each attribute, and the play Figurer in the final column counts the total number of occurrences of yes and no. In the lower part of the table, the same information was formed as fractions, or observed probabilities. For example, of

the nine days that play is yes, outlook is sunny for two, yielding a fraction of 2/9. For play the fractions are different: they are the proportion of days that play is yes and no, respectively.
Probability of yes = 2/ 9 × 3 /9×3/ 9×3/ 9×9 /14 = 0.0053.

The fractions are taken from the yes entries in the table according to the values of the attributes for the new day, and the final 9/14 is the overall fraction representing the proportion of days on which play is yes. A similar calculation for the outcome no leads to
Probability of no = 3/5×1/5×4/5×3/5×5/14 = 0.0206.

This simple and intuitive method is based on Bayes's rule of conditional probability. Bayes's rule says that if there is a hypothesis H and evidence E that bears on that hypothesis, then

$$\Pr(H|E) = \frac{\Pr(E|H)\Pr(H)}{\Pr(E)}$$

The notation Pr (A) denotes the probability of an event A and Pr (A|B) denotes the probability of A conditional on another event B. The hypothesis H is that play will be, say, yes, and Pr (H|E) is going to turn out to be 20.5%, just as determined previously. The evidence E is the particular combination of attribute values for the new day, *outlook = sunny*, *temperature = cool*, *humidity = high*, and *windy = true.*

3.4. Associative Classification

Classification rule mining and association rule mining are two important data mining techniques (Liu B., et al, 1998). The target of mining is not pre-determined for association rule mining, while for classification rule mining there is one and only one pre-determined target, i.e., the class.

Associative classification (Baralis, E., et al, 2002) proposed the integration of association rule mining and classification. Association rule is unsupervised learning that describes the co-occurrence among data items in a large amount of collected data (Agrawal and Srikant, 1994). Whereas, associative classification is a supervised task that predicts the class label of set data.

Differently from decision trees, associative classification uses association rules approaches that rely on the correspondence of values of different attributes, hence allowing one to achieve better accuracy. MCAR (Multi-class Classification based on Association Rule) (Thabtah F., et al, 2004) conducted many experimental studies and showed that associative classification is a

promising approach, which builds more accurate classifiers than traditional classification techniques such as C4.5.

Furthermore, with respect to other approaches (e.g., neural networks), the generated associative model is more understandable for a human being. Unfortunately, in large or highly correlated datasets, rule extraction algorithms adopted from association rules have to deal with the solution set (Toivonen, H., et al, 1995). This (i) makes the rule extraction process time consuming (and in some cases unfeasible), and (ii) makes it difficult to optimally exploit the generated rules.

To tackle the problem of generating large number of rules in association rule, different methods have been investigated. Some works toward pruning the discovered rules in order to form a small rule set (Toivonen, H., et al, 1995), other approaches discard rules that are less relevant with respect to statistical parameters such as support, confidence, and (Brin, S., et al, 1998). Other new developed methods have been proposed, as an alternative to rule pruning methods, to represent *ITEMSETs* and association rules in a compact form.

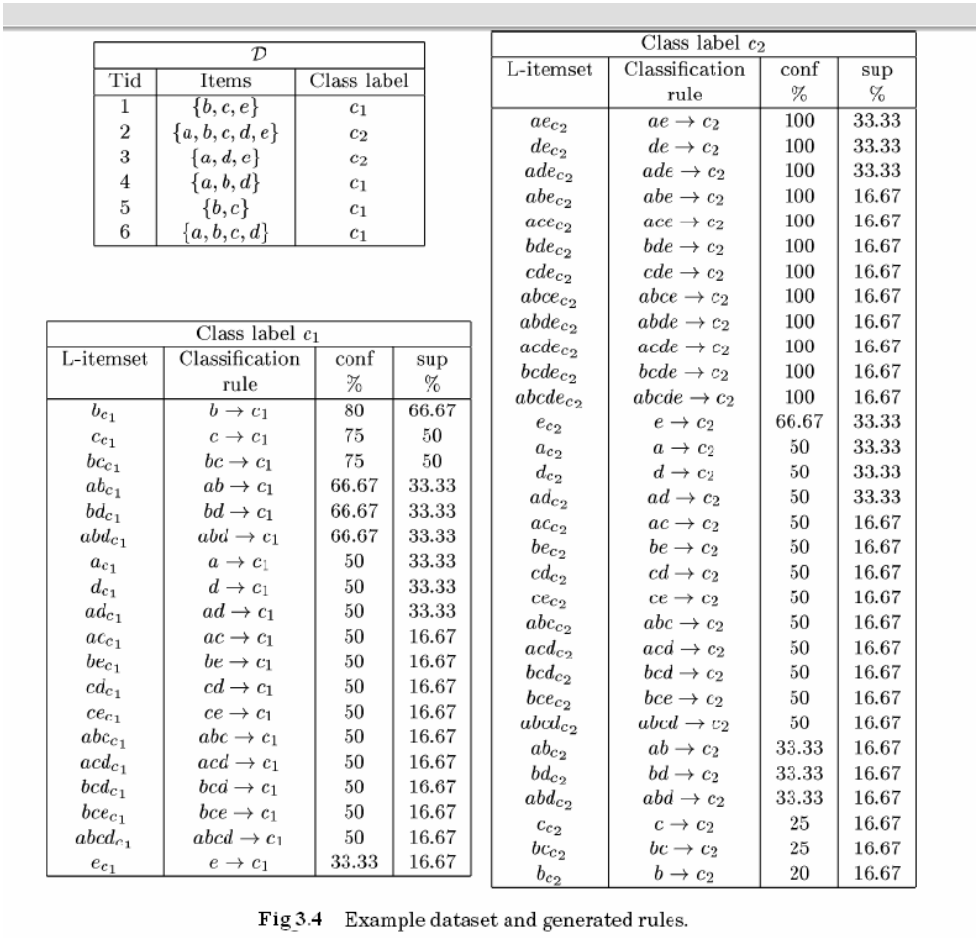
Several more effective compact forms have been proposed, which allow the generation of the complete rule set. In associative classification, the research focus has been mainly on pruning techniques. These techniques discard rules that either are redundant from a functional point of view, or may cause incorrect classification. Furthermore, they discard rules that achieve less accuracy in classification according to statistical parameters such as support, confidence, and chi-square test. Pruning is usually applied as a postprocessing step on the extracted rules. Only support based pruning is performed concurrently with the rule extraction process, since the downward closure property of support can be exploited to reduce the computational complexity.

3.4.1 Associative Classification Framework

This section introduces the associative classification problem. A training dataset R is characterized by a schema ($A1. . . A_k$, c), where ($A1, . . . , A_k$) are k distinct attributes and c is a class attribute. Each tuple in R is a data object, which is associated to a unique identifier called *tid*. The attributes may have either a categorical or a continuous domain. For categorical attributes, all values in the domain are mapped to consecutive positive integers. For continuous attributes, the value range is discretised into intervals, which are mapped into consecutive positive integers. In this way, all attributes are treated uniformly.

Each data object in *R* can be described as a collection of pairs *(attribute, integer value)*, plus a class label (a value belonging to the domain of the class attribute *c*). Each pair *(attribute, integer value)* is called *item* in the remaining part of the article.

In the following definition, the context for associative classification will be formalized.



$\mathcal{D}$

Tid	Items	Class label
1	$\{b, c, e\}$	c_1
2	$\{a, b, c, d, e\}$	c_2
3	$\{a, d, e\}$	c_2
4	$\{a, b, d\}$	c_1
5	$\{b, c\}$	c_1
6	$\{a, b, c, d\}$	c_1

Class label c_1

L-itemset	Classification rule	conf %	sup %
b_{c_1}	$b \rightarrow c_1$	80	66.67
c_{c_1}	$c \rightarrow c_1$	75	50
bc_{c_1}	$bc \rightarrow c_1$	75	50
ab_{c_1}	$ab \rightarrow c_1$	66.67	33.33
bd_{c_1}	$bd \rightarrow c_1$	66.67	33.33
abd_{c_1}	$abd \rightarrow c_1$	66.67	33.33
a_{c_1}	$a \rightarrow c_1$	50	33.33
d_{c_1}	$d \rightarrow c_1$	50	33.33
ad_{c_1}	$ad \rightarrow c_1$	50	33.33
ac_{c_1}	$ac \rightarrow c_1$	50	16.67
be_{c_1}	$be \rightarrow c_1$	50	16.67
cd_{c_1}	$cd \rightarrow c_1$	50	16.67
ce_{c_1}	$ce \rightarrow c_1$	50	16.67
abc_{c_1}	$abc \rightarrow c_1$	50	16.67
acd_{c_1}	$acd \rightarrow c_1$	50	16.67
bcd_{c_1}	$bcd \rightarrow c_1$	50	16.67
bce_{c_1}	$bce \rightarrow c_1$	50	16.67
$abcd_{c_1}$	$abcd \rightarrow c_1$	50	16.67
e_{c_1}	$e \rightarrow c_1$	33.33	16.67

Class label c_2

L-itemset	Classification rule	conf %	sup %
ae_{c_2}	$ae \rightarrow c_2$	100	33.33
de_{c_2}	$de \rightarrow c_2$	100	33.33
ade_{c_2}	$ade \rightarrow c_2$	100	33.33
abe_{c_2}	$abe \rightarrow c_2$	100	16.67
ace_{c_2}	$ace \rightarrow c_2$	100	16.67
bde_{c_2}	$bde \rightarrow c_2$	100	16.67
cde_{c_2}	$cde \rightarrow c_2$	100	16.67
$abce_{c_2}$	$abce \rightarrow c_2$	100	16.67
$abde_{c_2}$	$abde \rightarrow c_2$	100	16.67
$acde_{c_2}$	$acde \rightarrow c_2$	100	16.67
$bcde_{c_2}$	$bcde \rightarrow c_2$	100	16.67
$abcde_{c_2}$	$abcde \rightarrow c_2$	100	16.67
e_{c_2}	$e \rightarrow c_2$	66.67	33.33
a_{c_2}	$a \rightarrow c_2$	50	33.33
d_{c_2}	$d \rightarrow c_2$	50	33.33
ad_{c_2}	$ad \rightarrow c_2$	50	33.33
ac_{c_2}	$ac \rightarrow c_2$	50	16.67
be_{c_2}	$be \rightarrow c_2$	50	16.67
cd_{c_2}	$cd \rightarrow c_2$	50	16.67
ce_{c_2}	$ce \rightarrow c_2$	50	16.67
abc_{c_2}	$abc \rightarrow c_2$	50	16.67
acd_{c_2}	$acd \rightarrow c_2$	50	16.67
bcd_{c_2}	$bcd \rightarrow c_2$	50	16.67
bce_{c_2}	$bce \rightarrow c_2$	50	16.67
$abcd_{c_2}$	$abcd \rightarrow c_2$	50	16.67
ab_{c_2}	$ab \rightarrow c_2$	33.33	16.67
bd_{c_2}	$bd \rightarrow c_2$	33.33	16.67
abd_{c_2}	$abd \rightarrow c_2$	33.33	16.67
c_{c_2}	$c \rightarrow c_2$	25	16.67
bc_{c_2}	$bc \rightarrow c_2$	25	16.67
b_{c_2}	$b \rightarrow c_2$	20	16.67

Fig 3.4 Example dataset and generated rules.

A classification context will be also called a dataset. Figure 3.4 reports a small dataset used as a running example. The dataset includes six rows, five different items ($I = \{a, b, c, d, e\}$), and two different classes ($C = \{c1, c2\}$) which label the rows.

Definition 3.4 (*L-ITEMSET and L-TIDS-LIST*). Let $c \in C$ be an arbitrary class label. Let $X \subseteq 2^I$ be an arbitrary *ITEMSET* and $T \subseteq 2^T$ be an arbitrary *TIDS-LIST*. $Xc \subseteq 2^I \times C$ *is* a labeled *ITEMSET* (or l-*ITEMSET*) where $\forall i \in X$, c labels i. $Tc \subseteq 2^T \times C$ is a labeled *TIDS-LIST* (or l-*TIDS-LIST*) where $\forall t \in T$, c labels t.

The traditional concepts of *ITEMSET* and *TIDS-LIST* can be seen as a particular instance (i.e., an unlabeled instance) of the above definitions. They describe sets of items and sets of transactions in *D*, by ignoring the class label associated to the elements in the set. In the dataset in

Figure 3.4, bc_{c1} and ade_{c2} are l-*ITEMSETs*, and $\{1, 4\}_{c1}$ and $\{2, 3\}_{c2}$ are l-*TIDS-LIST*. *abcd* is an *ITEMSET*, and {1, 3, 4} is a *TIDS-LIST*, brackets were used to represent l-*TIDS-LIST* and *TIDS-LIST*.

The l-*ITEMSETs* mined from a dataset can be used to represent the classification rules that can be extracted from it. An arbitrary l-*ITEMSET* X_c in D encodes the *classification* rule $r : X \rightarrow c$, where X and c are the antecedent and consequent of r, respectively. Figure 3.4 shows the complete set of rules that can be extracted from the example dataset.

Accuracy is an important factor in assessing the success of data mining. When applied to data, accuracy refers to the rate of correct values in the data. When applied to models, accuracy refers to the degree of fit between the model and the data. This measures how error-free the model's predictions are. In associative classification, classification rules are used to model the most relevant properties characterizing classes of data, and to predict the class label for unknown (unlabeled) data. A classifier is a function from 2^I to C that allows the assignment of a class label to a data object. A classifier, able to predict the class label for data objects with high accuracy, is generated from a collection of transactions (i.e., data objects with a label, also called *training dataset*). In associative classification, the classifier is generated by selecting the most appropriate set of association rules. A rule $r : X \rightarrow c$ classifies or matches a data object d when $X \subseteq d$. In this case, rule r assigns class label c to data object d.

Several measures have been proposed to quantify the "interestingness" or the quality of an association (Yiming, M., 2000) (and a classification) rule. Frequently used quality indices are support and confidence (Agrawal and Srikant, 1994). In our setting, these measures correspond to the support and confidence of the l-*ITEMSET* encoding the classification rule. For an arbitrary classification rule *Xc*, the support *sup*(*Xc*) is the fraction of transactions in D which contain X and are labeled by class label c. For the rule antecedent X , the support *sup*(X) is the fraction of transactions in D which contain X .

The confidence of *Xc* is *conf* (*Xc*) = *sup*(*Xc*)/ *sup*(X). A l-*ITEMSET*, or an *ITEMSET*, is said to be *frequent* when its support is above a given frequency threshold, in the following denoted as *minsup*.

	X	$\neg X$	$\sum_{row}$
c	$sup(X_c)$	$sup(\neg X_c)$	$sup(c)$
$\neg c$	$sup(X_{\neg c})$	$sup(\neg X_{\neg c})$	$sup(\neg c)$
$\sum_{col}$	$sup(X)$	$sup(\neg X)$	$\|\mathcal{T}\|$

$$sup(\neg X) = |\mathcal{T}| - sup(X)$$
$$sup(\neg c) = |\mathcal{T}| - sup(c)$$
$$sup(\neg X_c) = sup(c) - sup(X_c)$$
$$sup(X_{\neg c}) = sup(X) - sup(X_c)$$
$$sup(\neg X_{\neg c}) = |\mathcal{T}| + sup(X_c) - sup(X) - sup(c)$$

Fig 3.5 2×2 contingency table $\mathcal{M}_{X_c}$ for classification rule X_c.

For a specific given classification rule X_c, these measures are defined by taking into account also the frequencies of other l-*ITEMSETs* in the dataset. In particular, the fraction of transactions: including *ITEMSET X* , but labeled with classes different from *c* (denoted as $X\neg_c$);

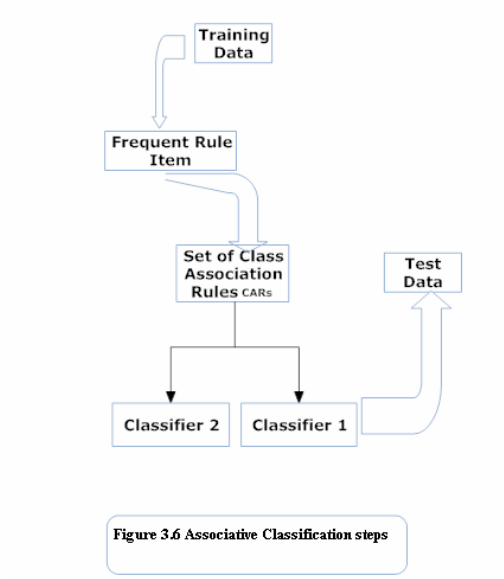


Figure 3.6 Associative Classification steps

not including *ITEMSET X* and labeled with class *c* (denoted as $\neg X_c$); not including *X* and labeled with classes different from *c* (denoted as $\neg X\neg_c$) are considered. Also the fraction of transactions labeled by class label *c* (denoted as *c*), not including *ITEMSET X* (denoted as $\neg X$), and labeled by a class label other than *c* (denoted as $\neg_c$), are considered.

These frequency counts can be tabulated in a 2 × 2 contingency table as shown in Figure 3.5. It can be easily seen that, for an arbitrary classification rule *Xc*, the content of each cell in the table can be expressed in terms of *sup*(*X*), *sup*(*c*), *sup*(X_C), and the number of transactions in the dataset (denoted $|T|$ in Figure 3.5.). Hence, for a rule X_C, any interestingness measure based on the frequency counts in the contingency table associated to X_C can be actually computed if *sup*(*X*), *sup*(*c*), *sup*(X_C), and $|T|$ are known. The contingency table for an arbitrary classification rule X_C *will* be denoted M_{XC}.

In general, the associative classification approach consists
of three major steps as Figure 3.6 shows:
1) generating frequent sets (i.e., com- bining pieces of evidence),

2) inducing strong rules.

3) ranking best rules and classification.

Although associative classification methods (Liu, B., et al, 2001) present several interesting aspects, they also suffer from some limitations. First, most of methods (Yin, X., et al, 2003) reported in the literature work under the *single-table assumption*, which is a strong limitation in those application domains characterized by a spatial dimension. Second, they have a categorical output which conveys no information on the potential uncertainty in classification. Small changes in the attribute values of an object being classified may result in sudden and inappropriate changes to the assigned class. Missing or imprecise information may prevent a new object from being classified at all.

3.4.2 CBA (Classification Based on Associations) Algorithm

According to (Liu, B., et al, 2001) the integration of association rule and classification is done by focusing on a special subset of association rules whose righthand- sides are restricted to the classification class attribute. It is refered to this subset of rules as the *class association rules* (CARs) (Liu, B., et al, 2001). An existing association rule mining algorithm (Agrawal and Srikant, 1994) is adapted to mine all the CARs that satisfy the minimum support and minimum confidence constraints. This adaptation is necessary for two main reasons:

1. Unlike a transactional database normally used in association rule mining (Agrawal and Srikant, 1994) that does not have many associations, classification data tends to contain a huge number of associations. Adaptation of the existing association rule mining algorithm to mine only the CARs is needed so as to reduce the number of rules generated, thus avoiding combinatorial explosion (see the evaluation section).

2. Classification datasets often contain many continuous (or numeric) attributes. The adaptation involves discretizing continuous attributes based on the classification pre-determined class target. There are many good discretization algorithms for this purpose (Fayyad U., 1996), (Dougherty, J., et al, 1995)

Data mining in the proposed associative classification framework consists of three steps (Yiming, M., et al, 2000):

- discretizing continuous attributes, if any
- generating all the class association rules (CARs), and
- building a classifier based on the generated CARs.

This work makes the following contributions:

1. It proposes a new way to build accurate classifiers. Experimental results show that classifiers built this way are, in general, more accurate than those produced by the state-of-the-art classification system C4.5 (Quinlan. J., 1993).

2. It makes association rule mining techniques applicable to classification tasks.

3. It helps to solve a number of important problems with the existing classification systems. The framework helps to solve the *understandability* problem (Clark, P., et al, 1993), (Pazzani, M., et al, 1997) in classification rule mining. Techniques such as those in (Liu, B., 1996), (Liu, B., 1997) can be employed to help the user identify understandable rules.

A related problem is the discovery of *interesting* or *useful* rules. The quest for a small set of rules of the existing classification systems results in many interesting and useful rules not being discovered. In this framework, the database can reside on disk rather than in the main memory. Standard classification systems need to load the entire database into the main memory (Quinlan. J., 1993), although some work has been done on the scaling up of classification systems (Mahta, M., et al, 1996).

CBA steps:

It consists of two parts (Liu, B., et al, 2001), a rule generator (called CBA-RG), which is based on algorithm Apriori for finding association rules in (Agrawal and Srikant, 1994), and a classifier builder (called CBA-CB).

CBA-RG algorithm: The CBA-RG algorithm (Liu, B., et al, 2001) generates all the frequent *ruleitems* by making multiple passes over the data. In the first pass, it counts the support of individual *ruleitem and* determines whether it is frequent. In each subsequent pass, it starts with the seed set of *ruleitems* found to be frequent in the previous pass. It uses this seed set to generate new possibly frequent *ruleitems*, called candidate *ruleitems*. The actual supports for these candidate *ruleitems* are calculated during the pass over the data. At the end of the pass, it determines which of the candidate *ruleitems* are actually frequent. From this set of frequent *ruleitems*, it produces the rules (*CARs*).

The CBA-RG algorithm is given in Figure 3.7 (Liu, B., et al, 1998).

```
1   F1 = {large 1-ruleitems};
2   CAR1 = genRules(F1);
3   prCAR1 = pruneRules(CAR1);
4   for (k = 2; Fk-1 ≠ ∅; k++) do
5       Ck = candidateGen(Fk-1);
6       for each data case d ∈ D do
7           Cd = ruleSubset(Ck, d);
8           for each candidate c ∈ Cd do
9               c.condsupCount++;
10              if d.class = c.class then c.rulesupCount++
11          end
12      end
13      Fk = {c ∈ Ck | c.rulesupCount ≥ minsup};
14      CARk = genRules(Fk);
15      prCARk = pruneRules(CARk);
16  end
17  CARs = ⋃k CARk;
18  prCARs = ⋃k prCARk;
```

Fig 3.7 : The CBA-RG algorithm

CBA-CB algorithm:

Have three steps:

Step 1: Sort the set of generated rules R according to the relation ">". This is to ensure that we will choose the highest precedence rules for our classifier as Figure 3.8 shows (Liu, B., et al, 2001).

Step 2: Select rules for the classifier from R following the sorted sequence, to be a potential rule classifier. Then compute and record the total number of errors that are made by the current class. This is the sum of the number of errors that have been made by all the selected rules and the number of errors to be made by the default class in the training data. When there is no rule or no training case left, the rule selection process is completed.

Step 3: Discard those rules that do not improve the accuracy of the classifier. The first rule at which there is the least number of errors recorded is the cutoff rule. All the rules after this rule can be discarded because they only produce more errors. The undiscarded rules and the default class of the last rule form the required classifier.

```
1   R = sort(R);
2   for each rule r ∈ R in sequence do
3       temp = ∅;
4       for each case d ∈ D do
5         if d satisfies the conditions of r then
6           store d.id in temp and mark r if it correctly
              classifies d;
7       if r is marked then
8           insert r at the end of C;
9           delete all the cases with the ids in temp from D;
10          selecting a default class for the current C;
11          compute the total number of errors of C;
12      end
13  end
14  Find the first rule p in C with the lowest total number
        of errors and drop all the rules after p in C;
15  Add the default class associated with p to end of C,
        and return C (our classifier).
```

Fig 3.8 Main algorithm for CBA-CB

3.4.2.1 The intersections of training objects locations

As we previously mentioned, there are two common representations of a target database; these are the horizontal (Agrawal and Srikant, 1994) and vertical (Zaki, M., et al, 2003) layouts. Table 3.4 illustrates a horizontal layout representation for a transactional database. In searching for frequent *ITEMSETs* in the horizontal layout, the database is scanned multiple times, once during each iteration, to perform support counting and pattern matching for candidate *ITEMSETs* at each level. Furthermore, computational overheads occur during support counting of candidate *ITEMSETs*.

According to (Zaki, M., et al, 2003), for each transaction with length *l*, during an iteration *n*, one needs to produce and evaluate whether all n-subsets of the transaction are contained in the current candidate list. In the vertical layout however, the database consists of a group of items where each item is followed by its *TIDS-LIST* (Savasere, A., et al, 1995) as shown in Table 3.5, which is a vertical representation of Table 3.4. A *TIDS-LIST* of an item is the transaction numbers (*TIDS-LIST*) in the database that contain that item.

Table 3.4 Horizontal representation of database

Tid	Items			
1	bread	milk	juice	
2	bread	juice	milk	
3	milk	beer	bread	juice
4	milk	eggs	milk	
5	beer	basket	bread	juice

Table 3.5 Vertical tid-list representation of database

Basket	Beer	Bread	Eggs	Juice	Milk
5	3	1	4	1	1
	5	2		2	2
		3		3	3
		5		5	4

Supports of frequent *ITEMSETs* are computed in the vertical layout by simple intersections of the *TIDS-LIST*. For instance, the supports of candidate *ITEMSETs* of size *k* can be easily obtained by intersecting the *TIDS-LIST*s of any two *(k-1)* subsets. The *TIDS-LISTs* that hold all the information related to items in the database are a relatively simple and easy to maintain data structure, and thus there is no need to scan the database during each iteration to obtain the supports of new candidate *ITEMSETs*, saving I/O time (Zaki, M., et al, 2003).

AC algorithms (Toivonen, H., et al, 1995) extend *TIDS-LISTs* intersections methods of vertical association rule data layout (Zaki, M., et al, 2003) to solve classification benchmark problems Supports of frequent *ITEMSETs* are computed in the vertical layout by simple intersections of the *TIDS-LIST*. For instance, the supports of candidate *ITEMSETs* of size *k* can be easily obtained by intersecting the *TIDS-LISTs* of any two *(k-1)* subsets. The *TIDS-LISTs* that hold all the information related to items in the database are a relatively simple and easy to maintain data structure, and thus there is no need to scan the database during each iteration to obtain the supports of new candidate *ITEMSETs*, saving I/O time (Zaki, M., et al, 2003).

3.4.2.2 Introducing Diffsets

Since each class is totally independent, in the sense that it has a list of all possible *ITEMSETs*, and their *TIDS-LIST*, that can be combined with each other to produce all frequent patterns sharing a class prefix, to avoid storing the entire *TIDS-LIST* of each member of a class. Instead

this method keeps track of only the differences in the *TIDS-LIST* between each class member and the class prefix *ITEMSET*. These differences in *TIDS-LIST* are stored in what we call the *diffset*, which is a difference of two *TIDS-LIST* (namely, the prefix *TIDS-LIST* and a class member's *TIDS-LIST*).

Using the *TIDS-LIST* for an *ITEMSET* in association rule discovery is a good approach as the cardinality of the *ITEMSET TIDS-LIST* divided by the total number of the transactions gives the support for that *ITEMSET*. However, the *TIDS-LIST* intersection methods presented in association rule discovery need to be modified in order to treat classification problems, where classes associated with each *ITEMSET* (rule antecedent) are considered when computing the support. the replacement of item covers in incidence matrices by their relative complement in its superpattern, so called *diffsets*, (Zaki, M., et al, 2003) developed a new approach called *dEclat* using the vertical database representation. They stored the difference of *TIDS-LIST* called *diffset* between a candidate k-*ITEMSET* and its prefix k-1- frequent *ITEMSETs*, instead of the *TIDS-LIST* intersection set, denoted here as *TIDS-LIST*.

A number of vertical mining algorithms have been proposed recently for association mining, which has shown to be very effective and usually outperform horizontal approaches. The main advantage of the vertical format is support for fast frequency counting via intersection operations on transaction ids (*TIDS-LIST*) and automatic pruning of irrelevant data.

In the vertical mining approaches there is usually no distinct candidate generation and support counting phase like in Apriori. Rather, counting is simultaneous with generation. For a given node or prefix class, one performs intersections of the *TIDS-LIST* of all pairs of class elements, and checks if min sup is met. Each resulting frequent *ITEMSET* is a class unto itself with its own elements that will be expanded in the next step.

Diffsets drastically cut down the size of memory required to store intermediate results. We show how *diffsets*, when incorporated into previous vertical mining methods, increase the performance significantly.

3.4.3 CMAR (Classification Based on Multiple Class-Association Rules)

This method extends an efficient frequent pattern mining method (Wenmin L., et al, 2001), FP-growth, constructs a class distribution-associated FP-tree, and mines large database efficiently. Moreover, it applies a CR-tree structure to store and retrieve mined association rules efficiently, and prunes rules effectively based on confidence, correlation and database coverage.

The classification is performed based on a weighted X^2 analysis using multiple strong association rules.

CMAR (Li W., et al, 2001), is consistent, highly effective at classification of various kinds of databases and has better average classification accuracy in comparison with CBA and C4.5. Instead of relying on a single rule for classification, CMAR determines the class label by a set of rules. To improve both accuracy and efficiency, CMAR employs a novel data structure, CR-tree, to compactly store and efficiently retrieve a large number of rules for classification. CR-tree is a prefix tree structure to explore the sharing among rules, which achieves substantial compactness. CR-tree itself is also an index structure for rules and serves rule retrieval efficiently. To speed up the mining of complete set of rules (Wenmin L., et al, 2001), CMAR adopts a variant of recently developed FP-growth method. FP-growth is much faster especially when there exist a huge number of rules, large training data sets, and long pattern rules.

The CMAR algorithm adopts the X^2 testing in its rules discovery step. When a rule is found, CMAR tests whether its body is positively correlated with the class. If a positive correlation is found, CMAR keeps the rule, otherwise the rule is discarded.

3.4.4 MCAR Multi-class Classification based on Association Rule

First, MCAR (Thabtah F., et al, 2005) scans the training data set once to count the occurrences of single items, from which it determines those that pass the *MinSupp* threshold. It stores items along with their locations (*TIDS*) inside arrays. Then, by intersecting the *TIDS* of the frequent items discovered so far, it can easily obtain the remaining frequent items that involve more than one attribute. It also uses *TIDS* for frequent single items to obtain support and confidence values for rules involving more than one item.

Once an item has been identified as a frequent item, the MCAR algorithm finds all rules with that item as condition which pass the *MinConf*. Considering that, only the rule with the largest confidence is counted by MCAR algorithm. In the case that an item has two rules with identical confidence, the choice of the rule will be random. MCAR always looks for the best rules for the final classification system.

The fact that training data set has been scanned only once to discover and generate the rules, makes this approach highly effective in runtime and storage since no multiple data scans are required. However in cases where there is large number of candidate items held in the main memory, the possible number of intersections required to generate frequent items is large.

3.4.5 Multi-label Classification

Single-label classification assigns an object to exactly one class, when there are two or more classes. Multi-label classification is the task of assigning an object simultaneously to one or multiple classes.

A novel approach for multi-label classification, was generated (Thabtah F., et al, 2004), is called multi-class, multi-label associative classification (MMAC). This technique assumes that for each training instance that passes certain thresholds, there is a rule associated with not only the most obvious class label, but with the second, third, k^{th} possible class labels.

MMAC (Thabtah F., et al, 2004) is an algorithm that follows the paradigm of associative

Input: Training data (*D*), *MinSupp* and *MinConf* thresholds
Output: A set of rules
Scan *D* for the set *S* of frequent single items
Do
    For each pair of disjoint items I_1, I_2 in *S*
    If $<I_1 \cup I_2>$ passes the *MinSupp* threshold
        $S \leftarrow S \cup <I_1 \cup I_2>$
Until no items which pass *MinSupp* are found
For each item *I* in *S*
    Generate all rules $I \rightarrow c$ which pass
        the *MinConf* threshold
Rank all rules generated
Remove all rules $I' \rightarrow c'$ from *S* where there is some rule $I \rightarrow c$ of a higher rank and $I \subseteq I'$.

Figure 3.9 MCAR algorithm

Classification, which deals with the construction of classification rule sets using association rule mining. MMAC learns an initial set of classification rules through association rule mining, removes the examples associated with this rule set and recursively learns a new rule set from the remaining examples until no further frequent items are left. These multiple rule sets might contain rules with similar preconditions but different labels on the right hand side. Such rules are merged into a single multi-label rule. The labels are ranked according to the support of the corresponding individual rules.

The algorithm consists of three phases: rules generation, recursive learning and classification. In the first phase, it scans the training data to discover and generate a complete CAR. In the second phase, MMAC proceeds to discover more rules that pass the *MinSupp* and *MinConf* thresholds from the remaining unclassified instances, until no further frequent items can

be found. In the third phase, the rules set derived from all iterations will be merged to form a global multi-class label classifier, which will be tested against test data. Figure 3.10 represents a general description of the method.

Input: Training data, confidence and support (σ) thresholds
Output: A set of multi-label rules and the classification accuracy
Phase 1:

- Scan the training data *T* with *n* columns to discover frequent items
- Produce a rules set by converting any frequent item that passes *MinConf* into a rule
- Rank the rules set according to (confidence, support, …, etc)
- Prune redundant rules from the rules set

Phase 2:

- Discard instances P_i associated with rules set just generated in phase 1.
- Generate new training data $T' \leftarrow T - P_i$
- Repeat phase 1 on T' until no further frequent item is found

Phase 3:

- Merge rules sets generated at each iteration to produce a multi-label classifier
- Classify test objects and calculate error rate using an accuracy measure

Figure 3.10 MMAC algorithm

3.5 Interesting Directions in Associative Classification

Constructing association rule discovery methods for classification systems in data mining is known as associative classification. In the last few years, associative classification algorithms such as CBA, CMAR and MMAC showed experimentally that they generate more accurate classifiers than traditional classification approaches such as decision trees and rule induction (Lim, T., et al, 2000). However, there is room to improve further the performance and/or the outcome quality of these algorithms (Thabtah, F., et al, 2006).

Associative classification is becoming a common approach in classification since it extracts very competitive classifiers with regards to prediction accuracy if compared with rule induction There are some challenges facing associative classification approach, which could improve solution quality and performance and also minimize drawbacks and limitations, such as incremental learning, noise in test data sets, exponential growth of rules and many others, probabilistic and decision tree approaches. However, challenges such as efficiency of rule discovery methods, the exponential growth of rules, rule ranking and noise in test data set need more consideration. Furthermore, there are new research directions in associative classification, which have not yet been explored such as incremental learning, multi-label classifiers and rules overlapping (Thabtah, F., et al, 2006).

3.6 Summary

In the beginning of this chapter, we surveyed different AC methods, and discussed their approaches used to find rules. Research work on AC to date is devoted to general classification problems where the aim is to build a classifier that contains single label rules. Most of AC algorithms aim to build accurate classifiers such as CBA, CMAR, MCAR and MMAC, these algorithms succeeded to build an accurate classifier, where only the most obvious class correlated to a rule is created and other classes are simply discarded. Sometimes the ignored class labels have frequencies in the training data above certain user thresholds, making their presence in the classifier important.

Also we referred to some challenges and interesting research Directions in Associative Classification approach, which should be considered in the future work.

CHAPTER 4

VERTICAL TEXT CATEGORIZATION (VTC)

4.1 Introduction

Text Categorization (TC), also known as Text Classification, is the task of automatically classifying a set of text documents into different categories from a predefined set (Sebastiani, F., 2002). If a document belongs to exactly one of the categories, it is a single-label classification task; otherwise, it is a multi-label classification task.

Associative classification (AC) (Baralis, E., et al, 2002) is the integration of association rule mining and classification. Association rule is unsupervised learning that describes the co-occurrence among data items in a large amount of collected data (Agrawal and Srikant, 1994). Whereas, associative classification is a supervised task that predicts the class label of test cases. Many studies show that AC frequently builds more accurate classifiers than traditional classification techniques, and that many of the rules found by AC methods can not be discovered by traditional classification algorithms (Thabtah F., et al, 2005). Also, classifiers generated by AC techniques contain rules that are easy to understand and can be manually altered by domain experts (Antonie, M., et al, 2002).

Data used is taken from Reuters-21578, Distribution 1.0" corpus, currently the most widely used benchmark in text categorization research. Reuters-21578 consists of a set of 12,902 news stories, partitioned (according to the "ModApte' split) into a training set of 9,603 documents and a test set of 3,299 documents, as well as different data sets from UCI data collection (Merz and Murphy, 1996).

Mining frequent patterns on the vertical data structures usually shows improvements of performance over the classical horizontal structure. This is because the vertical data structure supports fast frequency counting via intersection operations on transaction identifiers (*TIDS-LIST*). Recently, *Diffsets* (Zaki M., et al, 2001), a vertical data representation, has been introduced to improve memory requirement for intermediate TIDS-LIST storage in the mining process.

This thesis aims at finding better text classifiers along two of the following directions: (1) Using AC classifiers in order to improve the quality of the results with respect to classification accuracy, and (2) Employing the efficient method of vertical data format to represent the text documents in order to improve the efficiency of the classification model.

Most of the previous works on mining associations are based on the traditional horizontal transactional database format (Baralis, E., et al, 2002),8,12,13,15,16,18), a variant of Apriori-like Approaches, i.e. (1) (Baralis, E., et al, 2002) ( ) ( ), have utilized the horizontal data format, as

shown in Figure 4.1(Zaki M., et al, 2001), where DB = {1, 2, 3, 4, 5, 6}, and I = {A, B, C, D, E} is a set of five different items

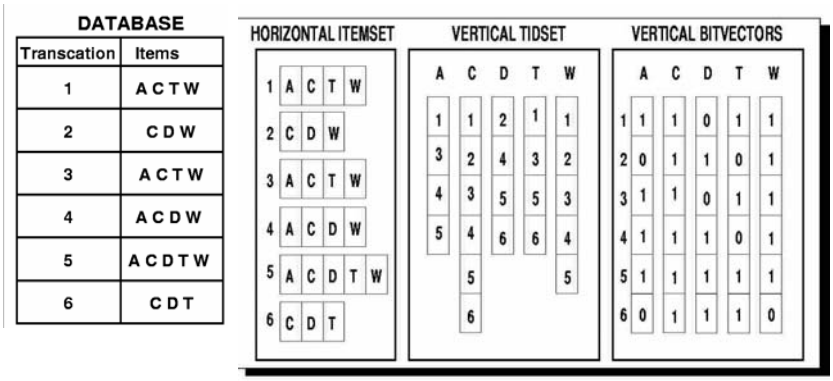


Figure 4.1 Horizontal and Vertical representation

in the database. Which represent the common horizontal data format that has been used often in mining associations. In this horizontal format, each transaction has a tid along with the ITEMSET comprising the transaction. However, recently a number of vertical mining algorithms such as VIPER (Shenoy, P., et al, 2000) and ECALT (Zaki M., et al, 2001) have been proposed for mining associations. In the vertical format, each item is associated with its corresponding *TIDS-LIST*, the set of all transactions (or *TIDS-LIST*) which contains that item as shown in Figure 4.1. Mining algorithms on the vertical format have shown to be very effective and usually outperform horizontal approaches (Agrawal and Srikant, 1994). This is since frequent patterns, which implies corresponding occurrence frequencies in the database for a given *ITEMSET*, can be counted via *TIDS-LIST* intersections, instead of using complex internal data structures Apriori candidate generation function (Agrawal and Srikant, 1994), which requires high computations. *Diffset* (Zaki M., et al, 2001) is a vertical data representation that keeps track only on the difference in the *TIDS-LIST* of a candidate pattern from its generating frequent patterns. It drastically cuts down the size of memory required to store intermediate results. The initial database stored in the format, instead of the *TIDS-LIST*, can also reduce the total database size.

Association mining works as follows. Let I be a set of items, and T a database of transactions, where each transaction has a unique identifier *(tid)* and contains a set of items. A set $X \subseteq I$ is also called an *ITEMSET*, and a set $Y \subseteq T$ is called a *TIDS-LIST*.

Using the *TIDS-LIST* for an *ITEMSET* in association rule discovery is a good approach as the cardinality of the *ITEMSET TIDS-LIST* divided by the total number of the transactions gives the support for that *ITEMSET*. However, the *TIDS-LIST* intersection methods presented in

association rule discovery need to be modified in order to treat classification problems, where classes associated with each *ITEMSET* (rule antecedent) are considered when computing the support. The replacement of item covers in incidence matrices by their relative complement, so called *diffsets*, (Zaki, M., et al, 2003) developed a new approach called *dEclat* using the vertical database representation. *dEcalt* stores the difference of *TIDS-LIST* called *diffset* between a candidate *k-ITEMSET* and its prefix *k-1- frequent ITEMSETs*, instead of the *TIDS-LIST* intersection set, denoted here as *TIDS-LIST*.

Our proposed algorithm deals with vertical mining and it is called the Vertical Text Categorization (VTC), which mines a complete set of frequent patterns within a given text on Diffset structure. The algorithm is instantiated using diffset structure based on AC approach to deal with text classification benchmark problems.

A common shortcoming in many TC applications is that it is expensive to classify data for the training phase, in order to learn a classifier that is able to correctly classify unseen documents. To deal with this shortcoming, it is necessary to train the TC algorithm with some pre-classified documents from each category, in such a way that the classifier is then able to generalize the model it has learned from the pre-classified documents and use that model to correctly classify the unseen documents.

For example, let's consider the SPAM DETECTION, Spams or, more formally, unsolicited commercial electronic messages, which can undermine the usability of electronic messages. Technical counter-measures include the development of spam filters, which can automatically detect a spam message. The problem is to classify if a given electronic message is spam or legitimate (a message is a data instance). Spams correspond to 25% of the total number of messages.

The organization of this chapter is as follows: Text Categorization problem is discussed in Section 4.2, and VTC is presented in Section 4.3. Section 4.4 points out general classification evaluation Metrics, and Section 4.5 discusses TC related evaluation methods. Data, preprocessing operation, and experimental results are presented in Sections 4.6, 4.7, and 4.8, respectively. Finally Section 4.9 is devoted to chapter summary.

4.2 Text Categorization Problem

The main goal of TC is to derive methods for the classification of natural language text (Sebastiani, F., 2002). The objective is to automatically derive methods that, given a set of training documents D = {*d1*, . . . , *dr*} with known categories C = {*c1*, . . . , *cq*} and a new

document q, which is usually called the query, will predict the query's category, that is, will associate with q one or more of the categories in C. The methods that are used in TC are generally the same that are used in the more general area of Information Retrieval (IR) or classification data mining, where the goal is to find relevant documents within a collection of documents that are related to a particular user query. By considering the document to classify as the query and the classes of the documents that are retrieved as the possible classes for the query, a method developed for IR can be used for TC tasks.

TC techniques are necessary to find relevant information in many different tasks that deal with large quantities of information in text form. Some of the most common tasks where these techniques are applied are: finding answers to similar questions that have been answered before; classifying news by subject or newsgroup; sorting spam from legitimate e-mail messages; finding Internet pages on a given subject. In each case, the goal is to assign the appropriate category or label to each document that needs to be classified.

4.2.1 Document Term Weighting

Document indexing is the process of mapping a document into a compact representation of its content that can be interpreted by a classifier. The techniques used to index documents in TC are borrowed from IR, where text documents are represented as a set of index terms which are weighted according to their importance for a particular document and for the collection in general (Salton G., 1968; Salton, G. 1971; Sebastiani, F., 2002; Yang, Li., 1999). In most of the current association classification methods, a shortage exists when these methods ignore the information about word's frequency in a text; this thesis presents a text categorization algorithm based on frequent pattern with term frequency.

Term frequency is important for text datasets, since it leads to higher performance, so it can't be ignored, especially to association rules defined based on probability of words occurrence (Koller, D., et al, 1997). A text document $\overrightarrow{d_j}$ is represented by an n-dimensional vector $\overrightarrow{d_j}$ of index terms or keywords, where each index term corresponds to a word that appears at least once in the initial text and has a weight associated to it, which should reflect how important this index term is. Regarding the problem of how to weight the terms in the documents, term weights can be binary-valued, indicating presence or absence of the term in the document; or real-valued, indicating the importance of the term in the document. There are multiple approaches for how real-valued weights can be computed. For example, (Sable, C., et al, 2001) introduces a bin-based term weighting method intended for tasks where there is insufficient training data to estimate a

separate weight for each word. (Sebastiani, F., et al, 2004b) proposes a supervised term weighting, where information on the membership of training documents to categories be used to determine term weights. However, none of the most recent approaches consistently outperforms the popular term weighting method proposed by (Salton, G., et al, 1988).

Formally, w_{ij}, the weight of term t_i for document $\overrightarrow{d_j}$, is defined as:

$$w_{ij} = \frac{freq_{ij}}{max_l(freq_{lj})} \times \log \frac{|D|}{n_{t_i}} \qquad (4.1)$$

For the reasons explained above, text documents are usually represented as a set of index terms which are weighted according to their importance for a particular document and for the collection in general, where the words in the document correspond to the index terms. The importance of each term, that is, its weight, can be computed in several ways, and the next sections describe the popular *tfidf*.

4.2.2 Term Frequency / Inverse Document Frequency

In the most usual case in TC, the weight *wij* of a term *ti* in a document $\overrightarrow{d_j}$ increases with the number of times that the term occurs in the document and decreases with the number of times the term occurs in the collection. This means that the importance of a term in a document is proportional to the number of times that the term appears in the document, while the importance of the term is inversely proportional to the number of times that the term appears in the entire collection. This term-weighting approach is referred to as term frequency/inverse document frequency (*tfidf*) (Salton and Buckley, 1988).

The mostly used weighting scheme in IR and TC method is the TFIDF (term frequency / inverse document frequency). TF(*w,d*) (Term Frequency) is the number of times word w occurs in a document, and d.DF(*w*) (Document Frequency) is the number of documents in which the word w occurs at least once.

The inverse document frequency is calculated as

$$IDF(w) = \log(\tfrac{|D|}{DF(w)}) \qquad (4.2)$$

Where *wi j*, the weight of term *ti* for document $\overrightarrow{d_j}$, is defined as:

$$w_{ij} = \frac{freq_{ij}}{max_l(freq_{lj})} \times \log \frac{|D|}{n_{t_i}} \quad (4.3)$$

Where $freq_{i\,j}$ is the number of times that term *ti* appears in document $\overrightarrow{d_j}$, |D| is the total number of documents in the collection, and nt_i is the number of documents where term t_i appears.

4.3 VTC Algorithm

Usually in searching for frequent *ITEMSETs* in the horizontal layout, the database is scanned multiple times, once during any iteration, to perform support counting for candidate *ITEMSETs* at each level. Furthermore, computational overheads occur during support counting of candidate *ITEMSETs*, according to (Zaki, M., et al, 2003), for each transaction with length *l*, during an iteration *n*. However In the vertical layout, the database consists of a group of items where each item is followed by its *TIDS-LIST* (Savasere, A., et al, 1995). A *TIDS-LIST* of an item is the transaction numbers (*TIDS-LIST*) in the database that contain that item.

Supports of frequent *ITEMSETs* are computed in the vertical layout by simple intersections of the *TIDS-LIST*. For instance, the supports of candidate *ITEMSETs* of size k can be easily obtained by intersecting the *TIDS-LISTs* of any two (k-1) subsets. The *TIDS-LISTs* that hold all the information related to items in the database are a relatively simple and easy to maintain data structure, and thus there is no need to scan the database during each iteration to obtain the supports of new candidate *ITEMSETs*, saving I/O time (Zaki, M., et al, 2003).

AC algorithms (Toivonen, H., et al, 1995) extend *TIDS-LISTs* intersections methods of vertical association rule data layout (Zaki, M., et al, 2003) to solve classification benchmark problems. We propose in this section a supervised learning method for text categorization called Vertical Text Categorization (VTC). During the Preprocessing phase, we introduce the terms frequent, and term frequency, and in the training phase, an extraction of the categories features is carried out. Then, the training data set is scanned once to discover frequent one-*ruleitems*, and then *ruleitem* with support and confidence larger than minsupp and minconf, respectively, is created as a potential rules. The next phase is called scoring and classification in which a given test data is assigned a class label based on the rules learned during the training phase. The proposed algorithm uses the vertical layout (Zaki, et al., 1997) for data representation and the fast

intersection method to discover the rules, as we will show below. Figure 4.2 shows the pseudocode of the proposed algorithm, which we will explain in details in section 4.3.2.

Figure 4.3 shows the training phase of the VTC, which we will explain in Section 4.2.2. Data used by VTC contain a header that indicates file name, attribute names, and a number of rows.

Input: Training data (D), min_supp, min_conf thresholds, the set of predefined categories $C = \{C1, \ldots, Cm\}$, where items start from i to j .
Output: A set of rules

Preprocessing phase
scan the training document *d*,
eliminate stopwords,
stemm words,
collect words and their *TF*s,
insert all these words into *D*.
Step 4: Convert data into the vertical diffset format using DiiffVTC procedure.
The Algorithm

Step 1: Find frequent items using the vertical data format
Step 2: Scan the database D and find the frequent 2-ITEMSETs and their supports using produce function.
Step 3: For each transaction T:

$R \leftarrow S_1$
$i \leftarrow 1$
while $(S_i \neq 0)$
{
$S_{i+1} \leftarrow produce(S_i)$
$R \leftarrow R \cup S_{i+1}$
$i \leftarrow i + 1$

}

Procedure DiffVTC
{
DiffVTC((S)):
for *all Xi* $\in$ *(S)* ***do***
for *all Xj* $\in$ *(S), with j > i* ***do***
$R = X_i \cup X_j$;
$d(R) = d(X_j) - d(X_i)$;
if $\sigma(R) \geq min\_sup$ ***then***
$T_i = Ti \cup \{R\}$;
for *all* $Ti \neq \phi$ ***do*** *DiffVTC*(T_i);
}

Fig. 4.2 VTC algorithm

Values for each training data instance are comma-separated, and the class attribute must be the last column in the header file.

Function produce
Input: A set of *ruleitems S*
Output: set of Ti produced ruleitems

$Ti \leftarrow 0$
Do
  For each pair of disjoint items X_i, X_j in *T* Do
    If (<$X_i \cup X_j$>, *c*) passes the *minsupp* threshold
      if (<$X_i \cup X_j$>, *c*) passes the *minconf* threshold
        $Ti \leftarrow Ti \cup (< X_i \cup X_j >, c)$

      end if
    end if
  end
end
Return Ti

Fig 4.3 The training algorithm of VTC

4.3.1 Training Data Format

According to (Agrawal and Srikant, 1994) and (Zaki, M., et al, 2003), the main advantage of the vertical format is support for fast frequency counting via intersection operations on transaction ids *(TIDS-LIST)* and automatic pruning of irrelevant data.

In the vertical mining approaches there is usually no distinct candidate generation and support counting phase like in Apriori. Rather, counting is simultaneous with generation. For a given node or prefix class, one performs intersections of the *TIDS-LIST* of all pairs of class elements, and checks if min sup is met. Each resulting frequent *ITEMSET* is a class unto itself with its own elements that will be expanded in the next step.

4.3.2 Frequent Ruleitems Discovery

To show how we find a frequent *ruleitem*, it is assumed that DB = {1, 2, 3, 4, 5, 6}, as shown in Table 4.1, and I = {FORECAST, BANK, ACCOUNT, OPER, MARKET } is a set of five different items in the database taken from Retuters 21578 dataset and was applied on VTC. Figure 4.4 depicts on the right part a common data format that has been used often in mining associations. In this horizontal format, each transaction has a tid along with the *ITEMSET* comprising the transaction. In contrast, the vertical format on the left part maintains for each item its TIDS-LIST, the set of all TIDS-LIST where the item occurs.

Figure 4.4(a) shows Diffset structure with support = 50%. In Figure 4.4(b), Diffset structure is sorted by the support in an ascending order and, hence, it has a better chance that more postfix

can be shared. Based on the example database in Table 4.1 and its diffset that arises in Figure 4.4, we will illustrate how to evaluate the support of *ITEMSETs* and, then, determine whether the *ITEMSETs* are frequent patterns.

It is assumed that the support threshold is specified as 50% which equals to 3. Now, let's examine a 2-*ITEMSET* (FORECAST,MARKET) first. We found that d(FORECAST,MARKET) = t(MARKET') - t(FORECAST') = {1, 3}, and the support of item FORECAST is σ(FORECAST) = 4. Consequently, the support of (FORECAST,MARKET) is σ(FORECAST,MARKET) = σ(FORECAST) - |d(FORECAST,MARKET)| = 4 - 2 = 2. *ITEMSET* FORECAST,MARKET is not a frequent pattern. If we evaluate a 3-*ITEMSET* (FORECAST,ACCOUNT,OPER), we will obtain that |d(FORECAST,ACCOUNT,OPER)| = |Ø| = 0, and σ (FORECAST,ACCOUNT) = 3. Hence, σ (FORECAST,ACCOUNT,OPER) = 3 - 0 = 3 which leads to a conclusion that FORECAST,ACCOUNT,OPER is a frequent pattern.

Table 4.1: Training Data

TID	Term
1	FORECAST, BANK, ACCOUNT, OPER
2	BANK, MARKET, OPER
3	FORECAST, BANK, ACCOUNT, OPER
4	FORECAST, BANK, MARKET, OPER
5	FORECAST, BANK, MARKET, ACCOUNT, OPER
6	BANK, MARKET, ACCOUNT

Horizontal Structure

1	FORECAST	BANK	ACCOUNT	OPER	
2	BANK	MARKET	OPER		
3	FORECAST	BANK	ACCOUNT	OPER	
4	FORECAST	BANK	MARKET	OPER	
5	FORECAST	BANK	MARKET	ACCOUNT	OPER
6	BANK	MARKET	ACCOUNT		

Vertical Structure

FORECAST	BANK	MARKET	ACCOUNT	OPER
1	1	2	1	1
3	2	4	3	2
4	3	5	5	3
5	4	6	6	4
	5			5
	6			

FORECAST	BANK	MARKET	ACCOUNT	OPER
2		1	2	6
6		3	4	

FORECAST	MARKET	ACCOUNT	OPER	BANK
2	1	2	6	
6	3	4		

Figure. 4.4(a) diffset for the training data in Table 4.1

Figure 4.5 (Zaki M., et al, 2001) shows how a typical vertical mining process would proceed from one class to the next using intersections of TIDS-LIST of frequent items. For example, the TIDS-LIST of FORECAST (t(FORECAST) = 1345) and of MARKET (t(MARKET) = 2456) can be intersected to get the TIDS-LIST for FORECAST,MARKET (t(FORECAST,MARKET) = 45) which is not frequent. Moreover, Figure 4.6 (Zaki M., et al, 2001) shows how diffsets can be used to enhance vertical mining methods. We can

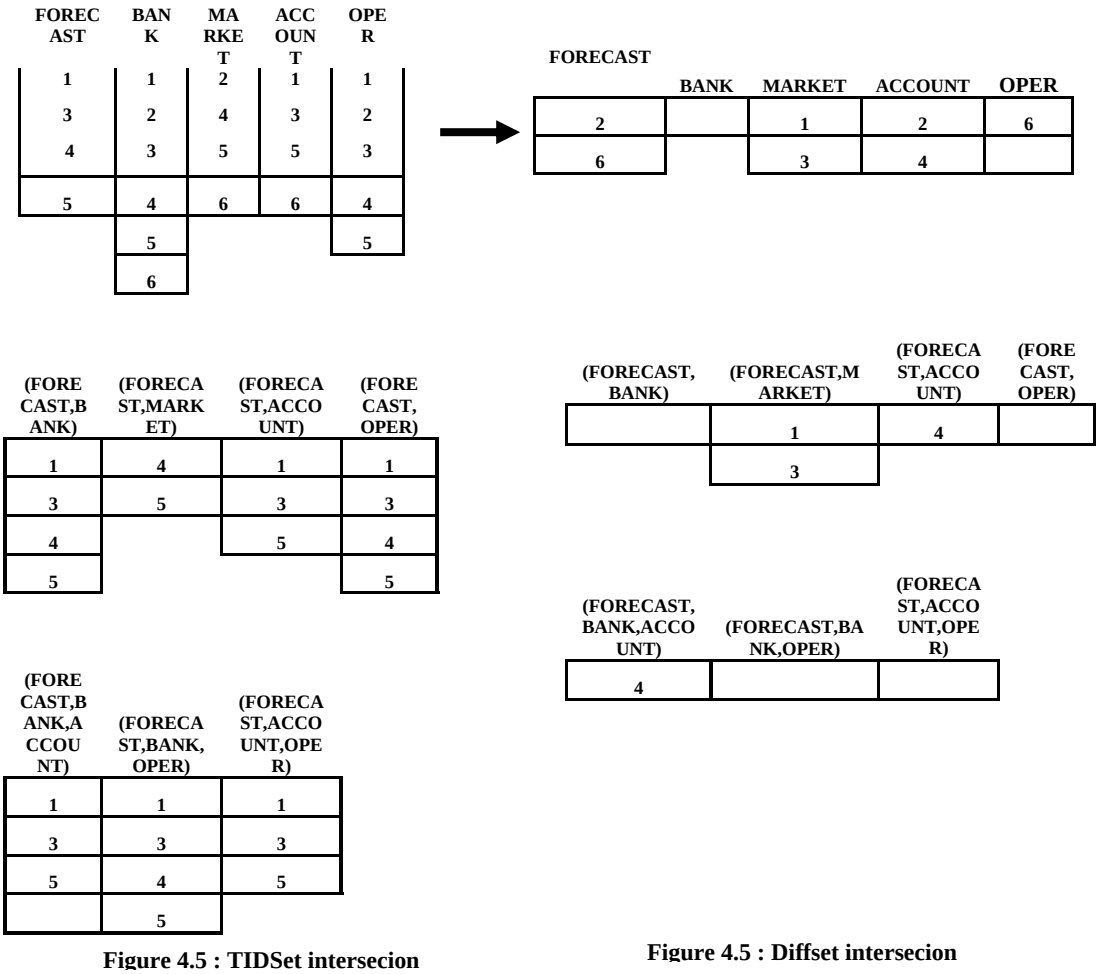


Figure 4.5 : TIDSet intersecion

Figure 4.5 : Diffset intersecion

start with the original set of TIDS-LIST for the frequent items, or we could convert the TIDS-LIST representation to a diffset representation at the very beginning. One can clearly observes that for dense datasets, a great reduction in the database size is achieved using this transformation.

The main operation used in the training phase of VTC is simple intersections between TIDS (TIDS-LISTs) of frequent *ITEMSETs*. There are no multiple database scans or candidate generation step, rather and during each iteration, only TIDS-LISTs of frequent *ITEMSETs* produced at the previous iteration are kept for further intersections. This reduces the amount of

information held in each iteration.To find frequent *ITEMSETs* at the current iteration, may lead to less memory use than keeping TIDS-LISTs for the whole frequent *ITEMSETs* generated in all previous iterations (Zaki M., et al, 2001).

4.3.3 Support and Confidence Computation and Rule Generation

In this section we briefly explain how support and confidence for *ruleitems* are calculated using an example and show how rules are generated. To find the support for a *ruleitem*, we use the TIDS-LIST of its *ITEMSET* to locate classes associated with it in the category array and select the category with the largest frequency. Then by taking the cardinality of the set of the TIDS where the *ITEMSET* and its largest category occur and dividing it by the size of the training data set, we can obtain the *ruleitem* support.

The calculation of the confidence is done similarly except that the denominator of the fraction is the size of the set of the TIDS of the *ruleitem* condition (its *ITEMSET*) instead of the size of the whole training data set. Frequent *ruleitems* are generated recursively from *ruleitems* conditions having a smaller number of attributes, starting from frequent one- *ruleitems* derived in a single pass through the training data set. It should be noted that every time a frequent *ruleitem* is found, only the rule with the largest confidence is considered. In the case that a *ruleitem* is associated with two classes with identical confidence, the choice of the rule is random.

Consider the vertical representation shown in Figure 4.3 earlier for the training data set shown in Table 4.1. Assume that minsupp and minconf have been set to 20% and 50%, respectively. During the scan, the frequent one-*ITEMSETs* that pass the minsupp threshold are identified, and all other infrequent *ITEMSETs* and their TIDS are discarded. Candidate two-*ITEMSETs*, which are produced by merging disjoint frequent one *ITEMSETs* are shown in Figure 4.5.

In order to avoid storing the entire *TIDS-LIST* of each member of a class, this algorithm keeps track of only the differences in the *TIDS-LIST* between each class member and the class prefix *ITEMSET* as we explained earlier. These differences in *TIDS-LIST* are stored in the *diffset*, which is a difference of two *TIDS-LISTs*; Figure 4.5 shows the *diffset* for our *ITEMSETs*. Once these *ITEMSETs* are identified, we check their supports and confidences simultaneously by locating classes that occur with their TIDS.

There is no separate phase to calculate the confidences for all frequent *ruleitems* in VTC, whereas the majority of current AC techniques (Liu, et al., 1998; Yin and Han, 2003; Baralis, et al., 2004; Antonie and Zaïane, 2004) produce frequent *ruleitems* in one step and find their confidences in a separate step.

4.4 Evaluation Metrics

The evaluation of the computational systems performance is often done in terms of the resources (time and space) they need to function. Text Classification mission is to classify a query document, by associating with it an ordered list of categories to which the query belongs, so it is not enough to classify a document as belonging to any set of categories in a reasonable amount of time. In addition, the categories should also be the correct ones, that is, the ones that the document in really belongs to.

The evaluation of a text categorization system is based on test samples that have been already labeled by human experts. For text categorization systems, the evaluation strategy used is inherited from traditional IR experience. David D. Lewis has an interested review on how evaluation is carried out in TC systems (Lewis, D., et al, 1992). The starting point is to compare between human-assigned key words and computer-assigned ones. Table 4.2 summarizes contingency four possible situations. Notice the subscript *i* at every value. That means that we compute those numbers for every class by looking the documents it has been assigned to, and the same could be done for every document by looking at assigned classes. In the following subsections, we survey popular evaluation measures used in IR and TC applications.

Table 4.2 Contingency Table

class c_i		assigned by expert? YES	NO
assigned by classifier?	YES	TP_i	FP_i
	NO	FN_i	TN_i

where

- TP_i *True positive.* Those assessments where system and human expert agree for a label assignment.
- FP_i *False positive.* Those labels assigned by the system that does not agree with expert assignment.
- FN_i *False negative.* Those labels the system failed to assign as they were by human expert.
- TN_i *True negative.* Those non assigned labels that also were discarded by the expert.

4.4.1 Accuracy

Accuracy is defined as the percentage of correctly classified documents from all documents which have been retrieved. It is generally used to evaluate single-label TC tasks (see, for instance, (Nigam et al, 2000; Han and Karypis, 2000; Chuang et al., 2000; Han et al., 2001; Lertnattee and Theeramunkong, 2004; Sebastiani, F., 2005)). Usually, Accuracy, which is shown in equation (4.4) is represented as a real value between 0 and 1.

$$Accuracy = \frac{\#\text{Correctly classified test documents}}{\#\text{Total test documents}} \quad (4.4)$$

4.4.2 Cross validation

A supervised learning algorithm needs labeled data to be trained and labeled data to be tested. Of course, these two sets must be disjoint to avoid a false estimation of performance, and this is the reason why multiple runs of the experiments are usually launched with a different partition at each turn. As soon as data has to be split into disjoint sets, one for training and another for testing, our results may differ depending on how we have chosen such partitions. For that, cross validation strategy tends to reduce the possible bias introduced by this process, we used a random seed when partitioning the training data in cross validation, by this random partitioning, and several times the same data set, it allow some items to be in each of the two sets. Thus, several partitions are made and the final result is an averaged measurement of the experiment over every partition made. Some studies, as pointed by Witten and Frank in their book (Witten, E., et al, 1999), estimate that 5 is a stable number for assuring a certain statistical independence when computing the evaluation values from the partitions done on the collection.

4.5 Other Evaluation Methods in Classification

AC techniques use an error-rate method (Witten and Frank, 2000), which is the opposite of accurate measure discussed above to evaluate the effectiveness of their classifiers (Liu, et al., 1998, Li, et al., 2001; Yin and Han; 2003). Using this method, the classifier simply predicts the class of a test data object, if it is correct, this will be counted as a success, other wise it will be counted as an error. The number of error cases divided by the total number of cases in a test data gives the overall error on this data. The error-rate of a classifier on a test data set measures its predictive accuracy.

Another evaluation method in classification applications such as text categorisation is precision, which has been originated with another method named recall in the Information Retrieval (IR) field by (Van, R., 1979). Precision and recall work as follow: one starts with a collection of objects/documents and has a query. Some of the objects pertain to the query and others do not. When objects are retrieved based on the query, we may make two kinds of

mistakes, false positives and false negatives. Precision measures the proportion of correct answers from all those that were retrieved. Recall measures the proportion of correct answerers retrieved from the set of all correct answers.

Generally and with respect to a given query, documents can be divided into four different sets as shown in Table 4.3 According to Table 4.3,

Table 4.3: Documents possible sets based on a query in IR

Iteration	Relevant	Irrelevant
Documents Retrieved	X	Y
Documents not Retrieved	Z	W

precision= $\frac{|X|}{|X \cup Y|}$

And recall= $\frac{|X|}{|X \cup Z|}$. For example, let's say someone has 5 blue and 7 red tickets in a set and he submitted a query to retrieve the blue ones. If he retrieves 6 tickets where 4 of them are blue and 2 that are red, it means that he got 4 out of 5 blue (1 false negative) and 2 red (2 false positives). Based on these results, precision=4/6 (4 blue out of 6 retrieved tickets), and recall= 4/5 (4 blue out of 5 in the initial set).

For classification problems in data mining, precision is similar to accuracy and we can look at this class by class or globally. For each class one can divide the number of correct classifications by the number of instances classified in that class to get precision. Globally, precision is the number of correct classifications divided by the total number of instances in the test set. Recall is better seen class by class, for a given class; one can divide the correct classifications by the number of instances that should have been classified in that class to obtain recall.

For multi-class and multi-label problems, methods such as precision and recall need to be combined in order to measure the performance of all classes. Therefore, a hybrid method, called F1 (Van, R., 1979), which measures the average effect of both precision and recall together, has been used in IR and text categorization. Overall, AC algorithms, including (Liu, et al., 1998; Baralis, et al,, 2000; Li, et al., 2001; Yin et al, 2003; Antonie, M., et al, 2004) use error-rate (accuracy) method to come up with the effectiveness of their classifiers. Using error-rate method to validate the predictive strength of classifier is not the optimum choice for multi-label classifiers, since only one class per rule contributes to overall effectiveness of the classifier.

4.5.1 Evaluation Methods Effect on Classification Data

4.5.1.1 Traditional Classification Data

The error-rate method considers only one class for each rule in computing the correct predictions and thus, it can be criticized for favoring only one class. Alternatively, label-weight assigns a value for each possible class in a rule according to its frequency in the training data. This gives the top ranked class in a rule the highest weight and not all the weight as error-rate method does. This method does not favor any class no matter what its ranking in a rule; instead it reflects the true distribution frequency for each class when associated with a particular *ITEMSET*.

4.5.1.2 Text Categorisation Data

Text categorization is a very effective way to organize enormous number of documents in Digital Libraries. Accurate classification of documents is able to not only enhance document search precision, but also facilitate browsing-by topic functionality. It is, nonetheless, difficult to obtain a satisfactory categorization accuracy compared to the corresponding results given by professional catalogers. This is due largely to the complexity of the pre-defined large-scaled category hierarchies that makes it difficult for learning algorithms to distinguish among categories.

In measuring the quality of a text categorizer, the test data collection as whole is normally divided into parts according to documents categories (class labels). Each document is evaluated in turn to identify whether or not it belongs to each category and the process is repeated for all available categories. In addition, methods like macro-averaging or micro-averaging (Yang, et al., 2002) can be utilized to summarize the values of recall (precision) for all available categories, aiming to derive the effectiveness of the classifier on the whole test data set.

4.6 Datasets

Datasets are collections of pre-classified documents. They are essential to develop and evaluate a TC system, that is, to train the system and then to test how well it behaves, when given a new document to classify. A dataset consists of a set of documents, along with the category or categories that each document belongs to. In a first step, called the training phase, some of these documents (called the training documents) are used to train the TC system, by allowing it to learn a model of the data. Afterwards, in a step called the test phase, the rest of the documents (called

the test documents) are used to test the TC system, to see how well the system behaves when classifying previously unseen documents.

In the TC field, the most commonly used collections are the 20-Newsgroups collection, the Reuters-21578 collection, and the Webkd collection, for this work, we choose the Reuters-21578 collection as well as UCI repository data to work on.

4.6.1 The Reuters-21578 Collection

One of the most widely examined text corpora from text classification is known as Reuters-21578, which comes from the Carnegie Group, Inc. and Reuters, Ltd. It is a collection of 21578 real-world news stories and news-agency headlines in the English language. The total dataset size is approximately 25 megabytes. Most of the stories are annotated with zero or more topics, according to their economic subject categories.

Other (orthogonal) annotations categories are present, such as people, places, organizations etc. Each of the annotation categories can be chosen for a prediction task, but topics is preferred in existing literature because it is more abstract. People, places and organization may likely be found when the corresponding name is spotted into the story text. Typically a document assigned to a category from one of these sets explicitly includes some form of the category name in the document's text. (Something which is usually not true for topics categories.) However, not all documents containing a named entity corresponding to the category name are assigned to this category, since the entity was required to be a focus of the news story (David D. Lewis, 1997). Thus these proper name categories are not as simple to assign correctly as might be thought.

All the documents contained in the Reuters-21578 collection appeared on the Reuters newswire and were manually classified. This collection is much skewed, with documents very unevenly distributed among different classes. The ModApté train/test split is generally used for classification tasks (Sebastiani, F., 2002).

Each text may be given one, more, or zero category labels. A "negative" example for a given category is a text for which that category has not been assigned. Text with no category labels assigned act as negative examples for all categories. As it is apparent from Figure 4.7, the majority of stories (47%) are a negative example for all categories. Almost all of the remaining texts have exactly one topic (44%), and the remaining 9% has two or more labels. In summary, the data set is unbalanced towards negative examples. The correctness of so many unlabelled documents is under question, so restricting training set to documents with at least one label might

be a good choice. Figure 4.7 shows the frequencies of the most frequent categories. The most represented category is topic earn, which was assigned to 17% of the assigned documents.

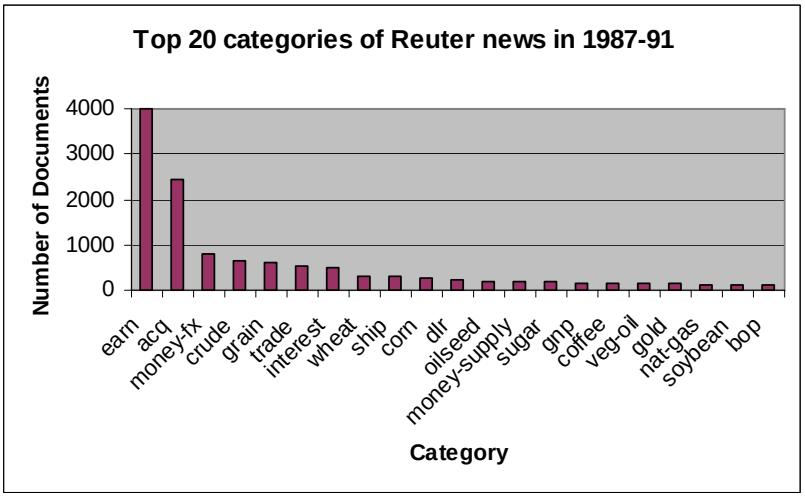


Figure 4.7 Reuters Categories

Reuters-21578 dataset from David Lewis' page (David D. Lewis,1997) used the standard "modApté" train/test split, the distribution of the documents per number of topics appears in the Table 4.4, here # train docs and # test docs refer to the Mod Apté split and # other refers to documents that were not considered in this split:

Table 4.4 The categories of Reuters-21578

Category	Number of training texts	Number of test texts
earn	2725	1051
acq	1490	644
money-fx	464	141
grain	399	135
crude	353	164
trade	339	133
interest	291	100
ship	197	87
wheat	199	66
corn	161	48

two sub-collections are usually considered for text categorization tasks (Debole and Sebastiani, F., 2004a):

• R10 – The set of documents belonging to the 10 classes with the highest number of positive training examples.

• R90 – The set of documents belonging to the 90 classes with at least one positive training and testing example.

Besides being much skewed, many of the documents in this collection are classified as having no topic at all or with more than one topic.

For the Reuters-21578 collections, from the original documents, the following pre-processing was applied:

1. Substitute TAB, NEWLINE, RETURN and punctuation characters by SPACE.
2. Substitute multiple SPACES by a single SPACE.
3. Turn all letters to lowercase.
4. Add the title/subject of each document in the beginning of the document's text.
5. Remove words that are less than 3 characters long.
6. Remove the 524 SMART stopwords. Some of them had already been removed, because they were shorter than 3 characters.
7. Apply Porter's Stemmer to the remaining words.
8. Feature weights were assigned using the *ltc* TF.IDF scheme (where *l* stands for logarithmic term frequency, *t* for logarithmic inverse document frequency, and *c* for cosine normalization).

All the files for the processed datasets are text files containing one document per line. Each document is composed by its class and its terms. Each document is represented by a "word" representing the document's class, a TAB character and then a sequence of "words" delimited by spaces, representing the terms contained in the document.

4.6.2 UCI Dataset.

Databases from UCI machine learning database repository (Merz, C., 1996), obtained from http://www.ics.uci.edu/_mlearn/MLRepository.html (UC-Irvine Machine Learning Data Repository), The University of California at Irvine (UCI) maintains a Machine Learning Repository of data sets for the development and testing of classification algorithms. It also maintains a Knowledge Discovery in Databases (KDD) Archive, an online repository of large data sets that encompasses a wide variety of data types, analysis tasks, and application areas. For information on these two repositories, see www.ics.uci.edu/~mlearn/MLRepository.html and http://kdd.ics.uci.edu.

4.7 Data Preprocessing.

As it is widely accepted that the way that documents and queries are represented influences the quality of the results that can be achieved. Keeping this fact in mind, there are several proposals that aim at improving retrieval results. The main aim of pre-processing the data is to reduce the problem's dimensionality by controlling the size of the system's vocabulary (the number of different index terms). In some situations, aside from reducing the complexity of the problem, this pre-processing will also make the data more uniform in a way that improves performance.

4.7.1 Stoplist Word Removal

Definition 4.1 (Stoplist) *Stoplist* is a list of words that are most frequent in a text corpus and are not discriminative of a message contents, such as prepositions, pronouns and conjunctions. Examples of stop words are "the", "and", "about", etc. Zipf's law is used to formulate a rule (Rij79) stating that the most frequent and the least frequent terms are usually not significant. *Stoplists* is one of the ways to eliminate the most frequent terms. Elimination of the least frequent terms is performed at the stage of rule generation by specifying a minimum support threshold. A set of words in a *stoplist* is very domain-specific. For example, the word "computer" can be discriminative in non-technical documents, but becomes a stop word in a corpus where all articles are dedicated to some computer science topic. Some sources suggest predefined *stoplists*, for example (Frakes, W., et al, 1992).

The *stoplist* used in the experiments contains 833 words and was obtained from the list in (Frakes, W., et al, 1992) by removing some words that turned out to be useful for the corpus and adding some others. The terms added are mostly those that are widely used in conversational and informal correspondence English, such as "I'm", "isn't", "asap", "thanks", "sorry", "can't", etc. The system provides a user with the interface to add/remove terms to/from the *stoplist*. Another way of satisfying Zipf's law would be to use corpus statistics and discard, say, 10% of the most frequent words. But this approach does not take into account a domain knowledge and can lead to the loss of words that are frequent by themselves but still quite informative if considered as part of a phrase.

4.7.2 Stemming

Definition 4.2 (Stemming) *Stemming* is the process of suffix removal to generate word stems. Although not always absolutely true, terms like "report", "reported" and "reports" do not make big difference for the purpose of distinguishing messages containing trip reports, for example, and can all be replaced by their stem "report". This increases support of the terms and thus avoids "losing" them when infrequent terms are filtered. Stemming also reduces feature space by almost 50% (Frakes, W., et al, 1992). Several different methods for automatic stemming are described in (Frakes, W., et al, 1992). One of them, Porter stemming algorithm, is used in the system.

4.8 VTC Analysis

Before starting our experiments, many steps had been made in order to prepare the Reuters documents for modeling, we summarize it as follows:

1. Documents were saved in txt file and indexed with an ID.
2. Training documents, whose NewID ranged according to the number of

Phases performed by VTC system are the following,

1. Preparation collection. Since we are in a Supervised Learning approach, we have to let our developed system learn from training data. Therefore, a collection of already indexed documents is provided.

2. Document representation. Documents are then processed and we find frequency for each term that occurs within the document.

3. Classifiers learning. For each class we train a classifier using documents labeled with such a class. After the training, a classifier per class is ready.

4. Testing/classification. Once the system is trained, we can either perform an automatic classification of new incoming documents or use an already labeled collection to test the performance of the trained system.

Our developed system implements cross-fold validation, and, therefore, the weighting of features and other operations cannot be performed before knowing the split of documents into training, validation and test sets:

- Training set: these documents are used for training the classifiers.
- Test set: these documents are used to compute the final performance of the system.

The amount of data available is enough to consider a k-fold cross validation methodology in the evaluation of different configurations for the experiments carried out. A cross validation (a

big part of the collection for training and the rest for testing) is performed k different times using a different partitioning at each turn into training and testing sets. The overall results are computed as an average of the subsequent k measurements registered.

Thus, these sets are produced in N different partitions (N being the number of folders used) which in our case has been set to 5. At every turn of the folding process, just one fold is used as test set, 1=5 of remaining folds as validation set, and the rest (4=5 of folds) is used as training set.

Once the system has been trained, it can assign labels automatically to new plain text documents. At each fold, values for different measures for the most frequent classes are given and the values of the averaged measures for the fold are shown. At the end, the averaged values of fold measures are computed.

For Reuters, which allowed the achievement of a high Accuracy using all 5485 training documents with VTC and SVM, Accuracy varies from 0.67 with one labeled document per class to 0.97 with 40 labeled documents per class for the VTC method.

In order to verify if the results obtained in the next section, 5-fold cross-validation tests were performed using all the datasets and the results were compared with the results obtained by each of the classification methods VTC, SVM, Naive Bayes, k-NN. The average is computed for the values of Accuracy for each of the 5 folds, and average Accuracy over all the folds for dataset and method.

Providing type of ranking classifiers makes the effects of thresholding strategies explicitly observable. A well-known baseline approach to category ranking, when categorical supervision is available, is to train a classifier independently for each class, and then to rank categories based on the confidence and support of the output of different classifiers.

The effects of thresholding strategies vary in different classifiers. In addition to varying cross validation randomization upon the given datasets, assigned for testing, it was noted that if two collections are statistically homogeneous, then the performance of a classifier should not vary appreciably between them; on the other hand, if the performance of a classifier changes dramatically when switching from a collection to a supposedly similar one, the differences between the collections is called for. Since the results of one classifier may be biased, we chose three different classifiers SVM, kNN and NBayes to evaluate the different versions of the Reuters collection. We chose these classifiers because they have fundamentally different classification algorithms, and we could closely control the conditions under which they were run, and support the same parameters and environment for testing same data sets.

4.8.1 VTC Experimental Results

In order to test the performance of our approach, we implemented the VTC algorithm in Java environment. In our experiments we have used the "Reuters-21578, Distribution 1.0" corpus, currently the most widely used benchmark in text categorization research. Reuters-21578 consists of a set of 12,902 news stories, partitioned (according to the "ModApte' split we have adopted) into a training set of 9,603 documents and a test set of 3,299 documents. We have discarded the categories that have no training examples, leaving us with 115 categories with at least one training example. We have also discarded all the (training and test) documents that have no label (originally, these documents were meant to be considered legitimate negative examples for all categories). This leaves us with a training set S consisting of 7,775 documents and a test set IT of 3,019 documents. The average number of categories per document is 1.08, ranging from 1 to 16; the number of positive examples per category ranges from 1 to 3964.

(VTC) were implemented in Java under Windows XP on a Pentium IV 3.2 Ghz, 896 RAM machine. Many studies have shown that the support threshold plays a major role in the overall classification accuracy of the set of rules produced by existing AC techniques (Liu, et al., 1998; Li, et al., 2001). The *minsupp* and *minconf* values were set to 3% and 30%, respectively, in the experiments. VTC algorithm and due to considering the term frequency in addition to applying AC thresholds, support and confidence measures, enhanced classification accuracy.

From our experiments, we observed that classifiers derived when the support threshold was set between 3% achieved good classification accuracy, and following the experiments, the *minsupp* was set to 3%. The confidence threshold, on the other hand, has a smaller impact on the behaviour of any TC method, and it has been set in our experiments to 30%. The results indicate that our proposed algorithm outperforms the other text categorization methods in terms of accuracy, which measured according to equation 4.1. Five-fold cross validation was used to derive the classifiers and error rates in the experiments.

The performance of association classification method is equal or sometimes higher than Bayes, KNN and SVM under lower feature numbers.

We presented experimental results with (Reuters). Results showed that improvements in average accuracy are performing well.

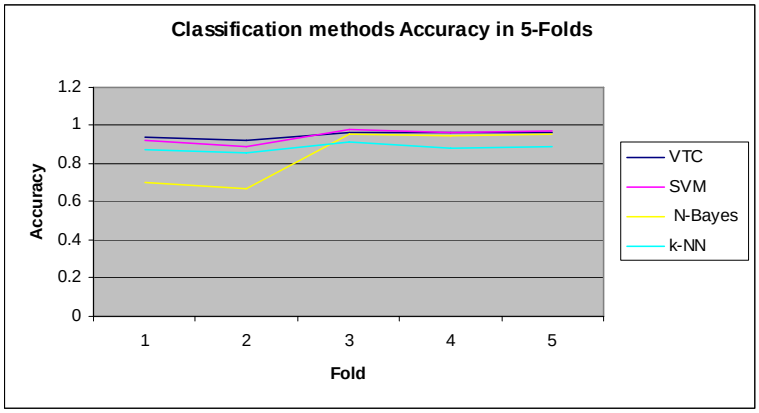


Figure 4.8

From the chart, it is obvious that for dataset, the algorithms underlying the SVM method generally work very well on very different datasets. By considering average Accuracy values, one can see that the VTC method is the second best, following the state-of-the-art SVM method for some data sets. After considering that the VTC method is accurate classifier, another important advantage of the VTC method is that it require a very small amount of memory to build the model of the data (only one vector to represent each class) during the training and test phases, as Figure 4.9 shows, due to using the vertical data representation, and *Diffset* structure, while others need to have all the training documents in memory at the same time to build the model of the data.

Average Size per Iteration: Tidset and Diffset

Database	*min_sup*	Max Length	Avg. Diffset Size	Avg. Tidset Size
Reuters 21578	0.5%	16	26	1820
	90%	12	143	62204
	5%	17	60	622
	35%	15	301	18977
	90%	8	330	45036
	0.025%	11	14	86
	0.1%	14	31	230
	0.5%	18	96	755

Figure 4.9 Average Tidset and Diffsets Cardinality

Table 4.5: Values of Accuracy for each of the classification methods VTC, SVM, Naive Bayes, k-NN, and Vector.

Dataset	Fold	VTC	SVM	N-Bayes	k-NN
Reuter 21578	1	0.93641	0.9245	0.705	0.8741
	2	0.92131	0.8921	0.6727	0.8561
	3	0.9635	0.9759	0.9557	0.9134
	4	0.9596	0.9661	0.9459	0.8853
	5	0.9611	0.9739	0.9531	0.8899

We used 5-fold cross-validation tests using all the datasets and compared the results obtained by each method. Table 4.5 contains the values of average Accuracy for each of the classification methods VTC, SVM, Naive Bayes, k-NN, and Vector, for each of the 5 folds, and average Accuracy over all the folds for each dataset and method as Figure 4.8 showed.

We also Experimented VTC on six different data sets from the UCI data collection (10) conducted using stratified ten-fold cross validation. The learning procedure is executed n times on slightly different training data sets.

Experiments on different data sets from UCI data collection (Balloon, Contact, Iris-Id, Led7, weather, and glass) were conducted. The experiments for both the CBA(Classification Based on Association) training step and our proposed algorithm with reference to the number of times *ITEMSETs* are merged in the training phase in each method. The *MinSupp* and *MinConf* used in the experiments were set to 5% and 40%, respectively The ultimate aim of the experiments is to compute the number of times *ITEMSETs* have been joined (merged) during each iteration in both CBA and our proposed method.

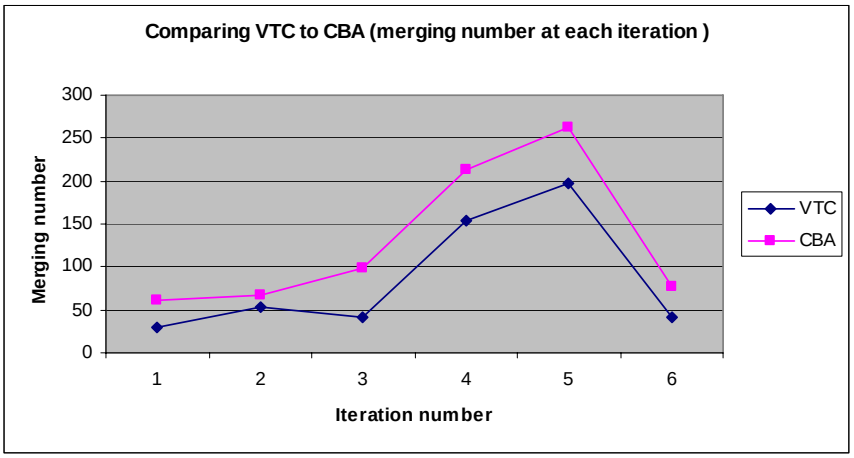


Figure 4.10 Comparing VTC to CBA(merging numbers)

It should be noted that we are only investigating the training phase (learning the rules) and not the classification step (building a classifier). Figure 4.10 shows the reduction of number of times *ITEMSETs* have been joined in each iteration for different classification benchmark problems, such as UCI data repository (Merz and Murphy, 1996) using the two approaches we consider, VTC and CBA. Particularly, we compute the number of times *ITEMSETs* have been merged at each iteration and for each data set we use. With our approach, the number of *ITEMSETs* that have been joined during each iteration is reduced significantly for Led7, Glassd,

Balloon, Contact, Iris-Id, weather, data sets. VTC has also reduced the number of joining in the training phase for the rest of the data sets.

4.9 Summary

Text Categorization is a research area that has provided efficient, effective, and working solutions that have been used in a variety of application domains. Two of the reasons for this success have been the involvement of the Machine Learning community in TC, which has resulted in the use of the very latest Machine Learning technology in TC applications, and the availability of standard data collections, which has encouraged research by providing a setting in which different research efforts can be compared to each other, so that the best methods can be discovered.

Currently, TC research is pointing in several interesting directions. One of them is to try to improve existing classifiers, by improving their effectiveness, or by making them faster to train or to test. Another is the attempt to find better representations for text.

In this chapter, we investigated the problem of generating rules using AC technique after applying *Diffset* structure to data representation to classification data. The results are a proposed approaches for text categorization VTC, which may result in higher classification accuracy for future instances. Data sets from Reuters data collection was tested indicated that VTC algorithm is effective, consistent and has higher classification accuracy than other algorithms.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

AC algorithms proposed in the literature usually adopt horizontal association rule mining methods to discover frequent *ruleitems*. In order to accomplish this task, these AC algorithms generally use the Apriori candidate generation or Frequent Pattern Growth approaches. However, these frequent *ruleitems* discovery approaches require multiple database scans, which necessitate high CPU time and large amount of data access. Alternatively and in order to improve the efficiency of frequent *ruleitems* discovery in AC, we present in this thesis a fast intersection method, which extends the vertical association rule mining approach to handle text categorization benchmark problems.

Our frequent *ruleitems* method has been used in the VTC algorithm, which finds rules in a single training data scan by performing fast intersections between frequent *ITEMSETs TIDS-LISTs*. Using a TIDS-LIST for an *ITEMSET* is a good approach since the cardinality of the *ITEMSET TIDS-LIST* divided by the total number of the transactions in the training data gives the support for that *ITEMSET*, then using *Diffset* structure which is a vertical data representation that keeps track only on the difference in the *TIDS-LIST* of a candidate pattern from its generating frequent patterns. It drastically cuts down the size of memory required to store intermediate results. The initial database stored in the format, instead of the *TIDS-LIST*, can also reduce the total database size.

Experimental results have showed that VTC is performing better results in term of accuracy on text categorization data sets and other classification benchmarks, comparing to other TC methods as SVM, KNN, and N.Bayes. The association categorization technique based on frequent patterns builds the classification rules by frequent patterns in various categories and classifies the new text employing these rules. However, in most of the current association classification methods, shortage exists when it is applied to classify text data, that these methods ignore the information about word's frequency in a text. In this thesis we consider term frequency probability of words occurrence, and enhance performance than other association categorization methods and some current text classification methods, since term frequency information of text is neglected in other text classification methods, based on association rules our method can achieve better efficiency even without rule pruning. Support is used for filtering out infrequent rules, while confidence measures the implication relationships from a set of items to one another.

This work also provided a comparison between the most used classification methods and other categorization methods. This comparison led to the conclusion that the use of AC can improve

the results obtained by the individual methods, even when they initially showed a good performance for some datasets, and that the improvement also depends on the difficulty level of the dataset that was used.

As there are many AC algorithms proposed in the literature that adopt horizontal association rule mining methods to discover frequent *ruleitems*. These AC algorithms generally use the Apriori candidate generation or Frequent Pattern Growth approaches to accomplish this task. However, these frequent *ruleitems* discovery approaches require multiple database scans, which necessitate high CPU time and large amount of data access. Alternatively and in order to improve the efficiency of frequent *ruleitems* discovery in AC, we present in this thesis a fast intersection method, which extends the vertical association rule mining approach to handle classification benchmark problems.

Our frequent *ruleitems* method has been proposed in the VTC algorithm, which finds rules in a single training data scan by performing fast intersections between frequent *ITEMSETs TIDS-LISTs*. Using a *TIDS-LIST* for an *ITEMSET* is a good approach since the cardinality of the *ITEMSET TIDS-LIST* divided by the total number of the transactions in the training data gives the support for that *ITEMSET*.

We compare the proposed algorithm with the CBA rule generation algorithm on six data sets from the UCI data repository in terms of database usage and run time. The proposed algorithm has shown good results, especially in terms of number of mergings in each iteration and execution times for almost all the data sets we consider. The physical memory usage is also reduced for most the data sets used in the experimental section. For future development, the VTC classifier approach will be tested and validated against further test data sets. This new approach of merging *ITEMSETs* can be used in most rule-based associative algorithms, to improve the execution times and to decrease the memory usage.

5.2 Future Work

In AC technique, the numbers of produced rules are still large; as a result, there is a need for new pruning methods in order to decrease the number of generated rules by AC techniques. Also a very active research area concerns Web, that refers to a second generation of web-based services in general not only documents which, among other things, allows internet users to publish documents that they find interesting along with classification keywords. The goal of these keywords is to facilitate searches on related topics. The classification methods studied in this work may be successfully applied to improve these searches. Moreover the presence of term in categorized documents is considered, we might study possibilities of absence of terms in the classification rules.

Usually, a classifier is constructed from labeled data records, and later is used to predict classes of previously unseen data as accurately as possible. Training and test data sets may contain noise, including, missing or incorrect values inside records. We should consider the importance of missing or incorrect values in test data sets in the prediction step. There have been some solutions to avoid noise in the training data sets. Naïve bayes for instance ignores missing values during the computation of probabilities, and thus missing values have no effect on the prediction since they have been omitted. Other classification techniques such as CBA treat them like other possible attribute values. However, the problem of dealing with noise in test data sets has not yet been explored well in AC. There are needs for developing processing methods that can handle test data with noise to effectively derive classifiers.

Selecting appropriate parameters to favour one rule on another in rule ordering is crucial task since most AC algorithms use rule ranking as the basis to select rules while constructing the classifier. The VTC algorithm favour rules principally with reference to confidence, support and lower cardinality. When several rules have identical confidence, support and cardinality, our method randomly choose one of the rules, which in some cases may degrade accuracy. Since AC approach generates normally large sized classifiers, where rules can be in the order of thousands, so that, there may be several rules with the same support, confidence and cardinality, for future work we might propose a rule ranking technique, to improved the accuracy of the resulting classifiers.

Another issue to be studied is the problem with Vertical format approach, when intermediate results of vertical tid lists become too large for memory, affecting the algorithm scalability.

REFERENCES

Agrawal R, H. Mannila, R. Srikant, H. Toivonen, and A. Inkeri Verkamo, "Fast *Discovery of Association Rules", Advances in Knowledge Discovery and Data Mining*", U. Fayyad and et al., eds., pp. 307±328, Menlo Park, Calif.: AAAI Press, 1996.

Agrawal, A. (1993) ' Mining association rules between sets of items in large databases', ACM SIGMOD.

Antonie, M. Zaïane, O (2004). An associative classifier based on positive and negative rules. Proceedings of the 9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery (pp. 64 - 69). Paris, France.

B. Dunkel and N. Soparkar. *Data organization and access for efficient data mining*. In 15th IEEE Intl. Conf. on Data Engineering, March 1999.

B. Liu, Y. Ma , R. Lee . *Analyzing the Interestingness of Association Rules from the Temporal Dimension*. IEEE, 2001.

B. Liu, Y. Ma , R. Lee . *Analyzing the Interestingness of Association Rules from the Temporal Dimension*. IEEE, 2001.

Baralis, E. And Garza, P. 2002. A lazy approach to pruning classification rules. In Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02). IEEE Computer Society Press, Los Alamitos, CA, 35–42.

Berger, A., Caruana, R., Cohn, D., Freitag, D., and Mittal, V. O. Bridging the lexical chasm: statistical approaches to answer-finding. In Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 192–199. Athens, Greece, 2000.

Brin, S.,Motwani, R., And Silverstein, C. 1998. Beyond market baskets: Generalizing association rules to dependence rules. Data Mining Knowl. Discov. J. 2, 1 (Jan.), 39–68.
C. Merz and P. Murphy. UCI Repository of Machine Learning Data- bases. Irvine, CA, University of California, Department of Information and Computer Science, 1996.

David D. Lewis, "Naïve (Bayes) at Forty: The Independence Assumption in Information Retrieval", in ECML-98, 1998.

Duda, R. And Hart, P. 1973. Pattern Classification and Scene Analysis. John Wiley and Sons, New York, NY.

F. Sebastiani. Machine learning in automated text categorization. ACM Computer Survey, 34(1):1–47, 2002.

Fayyad, U. M., Irani, K. B. "Multi-interval discretization of continuous valued attributes for classification learning" IJCAI-93, 1022-1027. Ghamrawi.N, McCallum.A, Collective Multi-Label Classification, CIKM'05, Germany.

Google Press Center. ”Google Achieves Search Milestone”. http://www.google.com/press/pressrel/6billion.html.

Huber, K.P., Berthold, M.R. (1995): Building precise classifiers with automatic rule extraction. IEEE International Conference on Neural Networks. 3, 1263–1268

Ian H. Witten, E. Frank. Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann Publishers, 2005.

J. Han and M. Kamber. ”Data Mining Concepts and Techniques”. Morgan Kaufmann Publishers, 2001.

J. Han, H. Pei, and Y. Yin. *"Mining Frequent Patterns without Candidate Generation*". In: Proc. Conf. on the Management of Data (SIGMOD’00, Dallas, TX). ACM Press, New York, NY, USA 2000.

J. Han, J. Pei, and Y. Yin, "Mining Frequent Patterns Without Candidate Generation," in Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data. Dallas,TX: ACM, 2000, pp. 1-12.

J. Shafer, R. Agrawal, and M. Mehta. Sprint: *A scalable parallel classifier for data mining*. In 22nd VLDB Conference, March 1996.

J.R. Quinlan. ”C4.5: Programms of Machine Learning”. Morgen Kaufmann, San Mateo, CA, 1993.

Joachims T. (1998). Text Categorization with Support Vector Machines: Learning with ManyRelevant Features. *Proceedings of the European Conference on Machine Learning* (*ECML*), (pp. 173-142). Berlin, 1998, Springer.

Koller, D. and Sahami, M. Hierarchically classifying documents using very few words. In Proceedings of ICML-97, 14th International Conference on Machine Learning, pages 170–178. Morgan Kaufmann Publishers, San Francisco, US, Nashville, US, 1997.

Le wis DD (1997) The reuters-21578 text categorization test collection, 1997.
http://kdd.ics.uci.edu/

Lewis, D. D. An evaluation of phrasal and clustered representations on a text categorization task. In Proceedings of SIGIR-92, 15th ACM International Conference on Research and Development in Information Retrieval, pages 37–50. ACM Press, New York, US, Kobenhavn, DK, 1992a.

Li, W. Han, J. Pei, J (2001) . CMAR: Accurate and efficient classification based on multiple-class association rule. In: ICDM’01, San Jose, CA, Nov. 2001.
Liu, B. and Hsu, W. 1996. “Post-analysis of learned rules” AAAI-96, 828-834.
Liu, B., Hsu, W. & Ma, Y. Integrating Classification and association rule mining. Proceedings of the KDD (pp. 80-86). New York, NY, 1998.

Liu, B., Hsu, W. and Chen, S. 1997. “Using general impressions to analyze discovered classification rules” KDD-97, 31-36.
Merz, C. J, and Murphy, P. UCI repository of machine learning database.
[http://www.cs.uci.edu/~mlearn/MLRepository.html].

M. J. Zaki and K. Gouda, Fast vertical mining using diffsets.Technical Report 01-1, Rensselaer Polytechnic Institute,USA, 2001.

M. J. Zaki, "Scalable Algorithms for Association Mining," IEEE Transactions on Knowledge and Data Engineering, vol. 12, pp. 372-390, 2000.

Ma Yiming, Data Mining with CBA’s Framework National University of Singapore 2000.

Pazzani, M., Mani, S. and Shankle, W. R. 1997. “Beyond concise and colorful: learning intelligible rules” KDD-97.

Quinlan, J.R. (1986): Induction of Decision Trees. Machine Learning. 6(1),81–106

Quinlan, J. (1987) Simplifying decision trees. International journal of man-machine studies, 27-(3), 221-248.

R. Agarwal, T. Imielinski, and A. Swami, "Mining Association Rules Between Sets of Items in Large Datasets," in Proceedings of the ACM SIGMOD Conference on Management of Data. Washington,D.C.: ACM, 1993, pp. 207-216.

R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In U. Fayyad and et al, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307–328. AAAI Press, Menlo Park, CA, 1996.

R. Elmasri and S. B. Navathe. *Fundamental of Database Systems* (4th ed.). Addison Wesley, 2003.

Rumelhart, D. E.,Hinton, G. E., Andwilliams, J. 1986. Learning internal representations by error propagation. In Parallel Distributed rocessing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations 1. MIT Press, Cambridge, MA, 318–362.

Savasere, A, Omiecinski, E and Navathe, S, 1995, an efficient algorithm for mining association rules in large databases. In Proceedings of the 21st conference on Very Large Databases (VLDB’95), Zurich, Switzerland, pp. 432–444.

Silberschatz, A., and Tuzhilin, A. “What makes patterns interesting in knowledge discovery systems.” IEEE Trans. on Know. And Data Eng. 8(6), 1996, 970-974.

Thabtah, F, Challenges and Interesting Research Directions in Associative Classification, Sixth IEEE International Conference on Data Mining - Workshops (ICDMW'06), Philadelphia University 2006.

Thabtah, F, Cowling, P and Peng, Y, 2004, MMAC: A new multi-class, multi-label associative classification approach. In Proceedings of the 4th IEEE International Conference on Data Mining (ICDM'04), Brighton, UK, pp. 217–224.

Thabtah, F, Cowling, P and Peng, Y, 2005, MCAR: Multi-class classification based on association rule approach. In Proceeding of the 3rd IEEE International Conference on Computer Systems and Applications, Cairo, Egypt, pp. 1–7.

Toivonen, H., Klemettinen, M., Ronkainen, P., Hatonen, K., and Manilla, H. 1995. Pruning and grouping discovered association rules. In workshop on Statistics, Machine Learning, and Knowledge Discovery in Databases (ECML'95). 47–52.

U. M. Fayyad and R. Uthurusamy (eds.). *Notes of AAAI'94 Workshop Knowledge Discovery in Databases (KDD'94)*. Seattle, WA, July 1994.

U.M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. "Knowledge Discovery and Data Mining: Towards a Unifying Framework". In Knowledge Discovery and Data Mining, pages 82–88, 1996.

Van Rijsbergan C. J., Information Retrieval, Butterworths, 1979.

W, Cohen. (1995) Fast effective rule induction. Proceedings of the 12th International Conference on Machine Learning, (pp. 115-123). Morgan Kaufmann, CA.

W. Li, J. Han, and J. Pei. Efficient classification based on multiple class-association rules. In Proc. of the Intl. Conf. on Data Mining, pages 369–376, 2001.

Wang J., "Data Mining with Computational Intelligenc," Springer, 2005.

Wang L, Fu. X. "Data Mining Data Mining Challenges and Opportunities," Idea Group, 2003.

Weiss, S. M., Apté, C., Damerau, F. J., Johnson, D. E., Oles, F. J., Goetz, T., and Hampp, T. Maximizing text-mining performance. IEEE Intelligent Systems, volume 14(4):pages 63–69, 1999.

WEKA : Data Mining Software in Java: http://www.cs.waikato.ac.nz/ml/weka.

Wenmin Li Jiawei Han Jian Pei, CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules, School of Computing Science, Simon Fraser University Burnaby, B.C., Canada 2001.

Wiener, E., Pedersen, J.O., Weigend, A.S. (1995). A neural network approach to topic spotting. Proceedings of the Fourth Annual Symposium on Document Analysis and Information Retrieval (SDAIR'95), (pp. 317-332).Las Vegas, Nevada, 1995.

Witten, I. Frank, E (2000). Data mining: practical Machine learning tools and techniques with Java implementations, San Francisco: Morgan Kaufmann.

X. Yin and J. Han. CPAR: Classification based on predictive association rules. In Proceedings 2003 SIAM International Conference on Data Mining (SDM'03), San Francisco, CA, May 2003.

Zaki, M and Gouda, K, 2003, fast vertical mining using diffsets. In Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, pp. 326–335.

APPENDIX A

Datasets

This appendix contains tables with the number of training documents, number of test documents and total number of documents per class for each dataset, for the train/test split used in my experiments. It also has a section about UCI data repository which we used within our work.

Reuters-21578 collection:

Table 4.4 The categories of Reuters-21578

Category	Number of training texts	Number of test texts
earn	2725	1051
acq	1490	644
money-fx	464	141
grain	399	135
crude	353	164
trade	339	133
interest	291	100
ship	197	87
wheat	199	66
corn	161	48

SAMPLE UCI DATA

BALLOON DATASE:

YELLOW,SMALL,STRETCH,ADULT,T

YELLOW,SMALL,STRETCH,ADULT,T

YELLOW,SMALL,STRETCH,CHILD,F

YELLOW,SMALL,DIP,ADULT,F

YELLOW,SMALL,DIP,CHILD,F

YELLOW,LARGE,STRETCH,ADULT,T

YELLOW,LARGE,STRETCH,ADULT,T

YELLOW,LARGE,STRETCH,CHILD,F

YELLOW,LARGE,DIP,ADULT,F

YELLOW,LARGE,DIP,CHILD,F

PURPLE,SMALL,STRETCH,ADULT,T

PURPLE,SMALL,STRETCH,ADULT,T

PURPLE,SMALL,STRETCH,CHILD,F

PURPLE,SMALL,DIP,ADULT,F

PURPLE,SMALL,DIP,CHILD,F

PURPLE,LARGE,STRETCH,ADULT,T

PURPLE,LARGE,STRETCH,ADULT,T

PURPLE,LARGE,STRETCH,CHILD,F

PURPLE,LARGE,DIP,ADULT,F