Rule pruning and prediction methods for Associative classification approach in data mining

Hussein Y. Abu Mansour

A thesis submitted to the University of Huddersfield in partial fulfilment of the requirements for the degree of Doctor of Philosophy

December 2012

DEDICATION

Finally this moment come as a result of couple of hard years. This thesis is devoted with innermost love and everlasting respect.
To my Father's soul, my mother, brothers and sisters. To my supervisors Dr. McCluskey and Dr. Thabtah, beloved fiends and to everyone who help me. Without their support and
Encouragement I could not have reached this phase

Hussein Mansour

TABLE OF CONTENT

LIST OF FIGURES

List of Tables

LIST OF ABBREVIATIONS

AC	Associative Classification
ACCF	Associative Classification Based on Closed Frequent Itemsets
ACCR	Association Classification Based on Compactness of Rules
ACN	Associative Classifier With Negative Rules
ARM	Associative Rule Mining
ARC-BC	Association Rule-Based Categorizer By Category
BCAR	Boosting Association Rules
CAR	Class Association Rule
CACA	Class Based Associative Classification Approach
CBA	Classification based on Association Rule
CMAR	Classification based on Multiple Class-Association Rules
CPAR	Classification based on Predictive Association Rules
DF	Document Frequency
DC	Dominant Class
FP	Frequent Pattern
FC	Full Coverage
FPC	Full & Partial Coverage
FPCC	Full & Partial Coverage Correctly
IDF	Inverse Document Frequency
IG	Information Gain
IR	Information Retrieval
IREP	Incremental Reduced Error Pruning
KNN	K-Nearest Neighbor
L^3	Live and Let Live
MCAR	Multi-class Classification based on Association Rule
MMAC	Multi-class, Multi-label Associative Classification

NB	Naïve Bayesian
NN	Neural Network
PC	Partial Coverage
PC^3	Partial Coverage Correctly Classify
RIPPER	Repeated Incremental Pruning to Produce Error Reduction
SVM	Support Vector Machine
TC	Text Categorisation
TF	Term Frequency
WIDF	Weighted Inverse Document Frequency

ABSTRACT

Recent studies in data mining revealed that Associative Classification (AC) data mining approach builds competitive classification classifiers with reference to accuracy when compared to classic classification approaches including decision tree and rule based. Nevertheless, AC algorithms suffer from a number of known defects as the generation of large number of rules which makes it hard for end-user to maintain and understand its outcome and the possible over-fitting issue caused by the confidence-based rule evaluation used by AC.

This thesis attempts to deal with above problems by presenting five new pruning methods, prediction method and employs them in an AC algorithm that significantly reduces the number of generated rules without having large impact on the prediction rate of the classifiers. Particularly, the new pruning methods that discard redundant and insignificant rules during building the classifier are employed. These pruning procedures remove any rule that either has no training case coverage or covers a training case without the requirement of class similarity between the rule class and that of the training case. This enables large coverage for each rule and reduces overfitting as well as construct accurate and moderated size classifiers. Beside, a novel class assignment method based on multiple rules is proposed which employs group of rule to make the prediction decision. The integration of both the pruning and prediction procedures has been used to enhanced a known AC algorithm called Multiple-class Classification based on Association Rules (MCAR) and resulted in competent model in regard to accuracy and classifier size called " Multiple-class Classification based on Association Rules 2 (MCAR2)". Experimental results against different datasets from the UCI data repository showed that the predictive power of the resulting classifiers in MCAR2 slightly increase and the resulting classifier size gets reduced comparing with other AC algorithms such as Multiple-class Classification based on Association Rules (MCAR).

CHAPTER ONE

INTRODUCTION

1.1 Motivation

The rapid evolution of technology in the computer industry have enabled people, companies and organizations to store a huge amount of data inside computers which in some cases ranged into terabytes in size which need a new approaches to deal with these data as well as process it. Data mining is an example on these data processing approaches.

Data mining approaches are advantageous in dense databases (Yin et al., 2003). Consider for instance a large retail business with a massive amount of purchasing transactions and customer's details, finding associations between customer’s different features can help the management people in making business related decisions. For example, if a marketing department in a retail store would like to lunch new sale on some goods that best reach their target customers, figuring out the correlations among the customers’ purchases behaviour as well as the customer’s attributes may help the managers to make such decision. In data mining context, these correlations are known as association rules, for example: 68% of the customers who purchase soft drinks are likely to purchase a chocolate as well. In the transactional database, suppose that those transactions that have both items (the soft drinks and chocolate) form 19% of the whole transactions size in the store database. The customers who purchase soft drinks represent the association rule’s antecedent and those who buy chocolates are known as an association rule’s consequent. The 68% of the association rule mentioned above denotes the strength of the rule and is known as rule’s confidence, while the 19% is a statistical significance measure, known as the rule’s support.

Classification on the other hand is the process of forming a classification model *cl* that maps a group of attributes to a class. This model is then used to forecast the classes of a new data based on the value of attributes.

Given a data set of historical transactions and customer's attributes, the problem is to discover the Class Association Rules (CARs) with significant supports and high confidences (attribute values that have frequencies above user specified minimum support and minimum confidence thresholds). A subset of the generated CARs is chosen to build a model (classifier) that could be used to predict the class labels of unseen data cases. This approach, which uses association rule to build classifiers, is called "associative classification" (AC). Unlike the classic classification approaches such as rule induction(Cohen, 1995) and decision trees(Quinlan, 1993) which usually construct small size classifiers, AC explores all associations between attribute values and their classes in the training data set aiming to construct larger size classifiers. This is because AC methods aim to produce additional useful knowledge missed by traditional methods which therefore should improve the classification accuracy within applications.

There is a wide range of profitable applications from data mining techniques beside the retail businesses and market basket such as credit card scoring, email classification, text categorization, digital library journals indexing and medical diagnosis.

The problems that can be evaluated by classification have an outcome that is affected by a set of indicator attributes. The basic objective is to estimate the effect of each indicator variable and its influence on the outcome. For example, a bank would have a historical data on borrower attributes such as job stability, credit history and income. Data mining could estimate the effect of each indicator variable on the ultimate outcome. These weights could be applied to future customer data to determine whether to grant a loan or reject the request. Further, in a digital library journal, there are large numbers of journals which belong to several categories; the process of assigning a journal to one or more applicable categories by a human requires effort, care and experience. An automated categorisation system that assigns journals based on their content to the correct category or set of categories could significantly reduce time, effort and error rate.

The problem of discovering the complete set of CARs requires substantial CPU time because of the requirement for multiple database passes. Hence, it is very essential to use an efficient method for rule discovery. Besides, cutting down the number of rules and keep the significant one may reduce the computations cost and increased the model efficiency.

According to several experimental studies (Liu et al., 1998) (Yin et al., 2003) (Thabtah et al., 2005), one of the main drawbacks of AC mining is that it often generates large number of rules since AC extract all the correlations among the items and the class are discovered as rules. The use of large number of rules necessitates high computation cost and often degrades the accuracy rates. Recent studies including Liu et al., 1998) (Yin et al., 2003) (Thabtah et al., 2010) believed that removing redundant and misleading rules that often lead to wrong classification might enhance model efficiency as well as effectiveness.

It has been also reported in some AC algorithms such as (Li et al., 2001) (Yin et al., 2003) that predictions procedure that based on one rule might degrade the classification accuracy and lead to favouring one rule in predicting the class label for the majority of test cases while some other classifications rules might be used, this is called prediction bias in classification. Yet, utilising group of rules in making prediction decision may slightly enhance the classification accuracy and prevent favouring one rule from predicting many test cases. Further discussions are presented in section 2.3.4.

1.2 Data mining

Data mining is one of the main phases in the knowledge discovery from database (KDD) which uses different data analysis tools to extracts useful patterns and knowledge from data (Agrawal et al., 1993). In this section, a brief overview on data mining, its main tasks and it’s domain of applications is given.

KDD process compromise many phases such as data selection, data cleansing, data reduction, pattern evaluation and visualisation of the discovered information where data mining is one of the main phases (Elmasri and Navathe, 1999). There are many tasks can be accomplished when utilising data mining approaches , including classification, clustering, association rule discovery and outlier analysis (Witten and Frank, 2000).

These tasks can be carried out using a range of data mining techniques that are adopted from different scientific areas such as statistics (Snedecor and Cochran, 1989), databases(Liu, et al., 1998) (Baralis, et al., 2004), probabilities (Quinlan, 1993) and artificial intelligence (Wiener et al., 1995). There is no single data mining technique applicable to all tasks and when it comes to choose a technique for a certain problem, the decision is very critical since one technique could work well for one problem and poor elsewhere. There are many criteria that can be considered before taking such a decision such as the size and nature of the data, attribute types (multimedia, text, real, etc), number of attributes, output format and more importantly the goal of application (Kuonen, 2004). The following sections describe the different data mining tasks:

1.2.1 Classification

Classification is the process of forming a model (classifier) from a historical data to guess the class label of an unseen data object. This model is derived by learning process by analysing a set of training set (cases whose class label is known). Common applications for classification including medical diagnoses (Soni et al., 2011), credit card scoring (Huang et al., 2007), websites type detection (Aburrous et al., 2010) and fraud detection (PHUA et al., 2010). There are wide range of classification approaches in the context of data mining, some of which are decision trees (Quinlan, 1993) such as C5.0, others are statistical based approaches such as naive Bayesian (Holte, 1993), k-nearest Neighbour (Yang, 1999) and support vector machines (Vapnik, 1995), Rule indication approach such as IRIP (Furnkranz and Widmer, 1994) and RIPPER (Cohen, 1995).

1.2.2 Associations Rule Mining

Association rule mining is the process of discovering the patterns that occur frequently in a data such as the *frequent items* in a customer shopping cart. *Frequent items* refer to the set of items that occurs together frequently in a transactional database (Agrawal R. and Srikant, 1994). These frequent items are employed to generate the set of association rules. In other words, the association rules simply describe a shopping behaviour of customers in retail stores. Items are considered frequent if they occurs in the database for a certain times greater than or equal a predefined thresholds called Minimum Support.

1.2.3 Clustering

Clustering approaches analyse data objects without knowing their class labels. A set of objects are grouped according to a certain criterion such as the similarity among objects. Each object in a cluster is correlated with other objects in the same cluster (homogeneous objects). Each cluster can be viewed as a class of objects and then the rules can be derived from each cluster. Market segmentation for identifying common characteristic for groups of people is a good example of application where clustering can be employed (Rui Xu. 2005).

1.2.4 Regression

Regression is a statistical analysis often used to model and analyse several variables and it often used for numerical data prediction (Fayyad et al., 1996). Regression is a special case of classification. Regression can be presented in many formats, some of them are: 1) Liner regression, this can be used when the relationship between the predictors and the target can be estimated with a straight line. 2) Nonlinear Regression, in some cases the relations between two parameters can't be estimated as a striate line. In this case the nonlinear regression is used by pre-processing the data to have a linear relationship. 3)Multivariate Linear Regression, this refer to two or more indicators, here the regression lines cannot be visualized in two dimensional rather, each line can be com0iuted by extending the equation of a single predictor i.e. Liner to include the parameters for each other predictors.

Regression models are often tested by computing different statistics that determine the difference between the predicted values and the expected ones. Regression approach can be employed in many applications in business planning, marketing, time series prediction, financial forecasting, biomedical and drug response modelling, and environmental modelling. (Docs.Oracle.com)

1.3 Associative Classification (AC)

AC is a branch of study of data mining (Liu, et al., 2001). AC approach has been proposed and successfully employed to form classifiers (Liu et al., 1998). This study has attracted extensive research works from the knowledge discovery and machine learning communities including (Li et al., 2001) (Yin & Han, 2003) (Thabtah et al., 2005) (Li X. et al., 2008) (Chen et al., 2012). AC is a promising approach that uses association rule mining in forming its classifier that would be used to predict the class label for the unseen data. It compromise advantages from fields, association rule mining and classification. Further details about AC will be presented in Chapter 2. This section defines the AC problem, and discusses the potential solution scheme.

1.3.1 AC Problem Statement

Definition 1: A row or a training case in *D* can be described as a combination of attributes Ai and values a_{ij}, and a class denoted by c_j.

Definition 2: An attribute value can be described as a term name A_i and a value a_i, denoted <(A_i, a_i)>.

Definition 3: An *AttributeValueSet* can be described as a set of disjoint attribute values contained in a training case, denoted < (A_{i1}, a_{i1}), …, (A_{ij}, a_{ij})>.

Definition 4: A ruleitem *r* is of the form < AttributeValueSet, c>, where c$\in$C is the class.

Definition 5: A ruleitem r passes the minsupp threshold if $(AVSFreq(r)/|D|) \geq minsupp$,

Definition 6: A ruleitem r passes the minconf threshold if $(RIFreq(r)/\ AVSFreq\ (r)) \geq$ minconf.

Definition 7: Any ruleitem r that passes the minsupp threshold is said to be a frequent ruleitem.

Consider a training dataset *D* which contains *I* as a set of items (attribute values), and *C* as a set of classes. *d* is a data case in *D* where $d \in D$ that is presented by a set of attribute values. A *ruleitem* is the form of < (*Attributevalueset*), *c*> where $Attributevalueset \subseteq I$, and representing a set of attribute values, i.e. $(A_{i1}, a_{i1}),...,(A_{ij}, a_{ij})$, and *c* is a class. *Attributevalueset* (*AVS*) frequency (*AVSFreq*) is the number of tuples in *D* that matches the *AVS*. The *ruleitems* frequency (*RIFreq*) is the number of tuples in *D* that matches the *ruleitems* body within the same class of the *ruleitem*. *minsupp* is a user predefined threshold that judges whether *a ruleitem* is good enough (frequent) or not. The first step in any AC algorithm is to discover the complete set of frequent *ruleitems* (those which has a frequency larger than or equal to the *minsupp* threshold*).*

A rule *R* in the classifier has the form $(A_{i1}, a_{i1}) \wedge ... \wedge (A_{ij}, a_{ij}) \rightarrow c$ where the antecedent is conjunction of disjoint *Attributevalueset,* and *c* is a class. The set of rules is produced from the frequent *ruleitems* and represent the *ruleitemset* that pass the *minconf* threshold. In other words, a frequent *ruleitem* becomes a rule if its frequency (*RIFreq*) divided by the *AVSFreq* is larger than the *minconf.* The ultimate aim of AC algorithms is to extract the complete set of rules that satisfy the *minsupp* and *minconf* thresholds in order to build the classifier which is utilised to forecast test cases.

1.3.2 Solution Scheme

AC intends to achieve two goals, firstly to generate a set of rules that survive the *minsupp* and *minconf* threshold starting by scanning the dataset to find the set of the frequent *ruleitems*. The set of rules are then generated from these frequent *ruleitems*, and then a punning procedure will be invoked to evaluate the set of generated rules. Secondly it selects a significant subset of rules generated in step one to construct the classifier *Cl* for predicting the class labels of previously unseen cases. Consider for example the training dataset given in Table 1.1

Table 1.1: Example of a training data

TID	Att1	Att2	Class
1	*C*	*T*	cl_1
2	*C*	*X*	cl_2
3	*C*	*T*	cl_2
4	*C*	*X*	cl_1
5	*D*	*T*	cl_2
6	*D*	*T*	cl_1
7	*D*	*Y*	cl_1
8	*C*	*Y*	cl_1
9	*D*	*Z*	cl_1
10	*E*	*T*	cl_1

The first step in AC approach is to discover the frequent *ruleitems* starting by the single *ruleitems* (frequent 1- *ruleitems*) i.e. those that consists of only a single attribute value. Frequent one *ruleitems* are used for the discovery of potential frequent two *ruleitems*, and frequent two *ruleitems* are the input for the discovery of potential frequent three *ruleitems* and so forth. According to Table 1.1, and with *minsupp* 30%, the frequent one *ruleitems* set is: < *(Att1, C), cl*$_1$>, < *(Att1, D), cl*$_1$>, < *(Att2, T), cl*$_1$>. The disjoint between them results in the frequent *two ruleitems* which is ∅ in this example.

1.4 Research Aims, Scope and Objectives

This thesis aims to accomplish a number of aims, as a first aim, the thesis has the tendency to produce an extensive literature review on common AC approaches with more attention paid on two important phases, rule pruning and class assignment phases. Approaches used in both phases have been discussed in details so the reader can extract some future trends in AC. These two phases have been discussed due to their importance in solving the two main deficiencies in AC approach; the large number of produced Class Association Rule (CARs) and the overfitting problems. The thesis also aims to investigate the impact of cutting down the number of rules on the model efficiency and effectiveness Minimising the classifier size will done through a number of pruning procedures that evaluate the rule based on its coverage power with or without class correctness.

Most of the current AC algorithms are employing single high confidence rule approach for class assignment task, the thesis aims to discuss this approach, analyse it and examine the impact of employing multiple rules to predict classes for test cases on the classification accuracy. Besides, the thesis aims to develop an AC model by employing novel pruning and class assignment procedures. Furthermore, Text Categorisation (TC) problem will be exploited and its main phases will be discussed. The developed AC model at the previous stage will be adapted to TC problem and compared with other TC classifiers form AC and classical classification methods.
Lastly, after achieving the above mentioned aims, the thesis will answer a number of research questions:

1. If a rule is considered significant when its body is fully match a training case body (left right side) regardless to the class correctness (Matching between the rule's class and the training case class) during the classifier learning step, does the accuracy rate affected?
2. If a rule is considered significant when its body is partially match a training case during the classifier construction, does the accuracy affected?

3. When employing more than one rule (multiple rules) to make prediction decisions, does the accuracy positively affected?

4. Although AC often produces large number of rules, does the effectiveness and efficiency positively affected when adapting AC to TC?

1.5 Thesis Contributions

There are several achievements in this research work including the development of five pruning, and one prediction methods. Another important contribution in this thesis is the dissemination of an AC algorithm applicable to TC problem and deals with both structured and unstructured data. These issues and others are addressed in the following subsections.

1.5.3 New five rule pruning methods

AC adopts association rule mining to discover frequent ruleitems and generate a set of candidate rules. Association rules considers all correlation between items in a database since the objects in a dataset are often highly correlated, the expected number of Class association rules is often enormous; ranged into thousands, or even hundreds of thousands when dealing with dense data like text data. Many of the produced rules are redundant, misleading or conflicting with other rules. Not all of the rules derived during the learning phase can be used to form the classifier. Hence, triggering pruning procedures including Pre-pruning (Pruning before generating the set of rules) and post-pruning (during the rules generation) become essential to enhance the generated rules and filter out such uninteresting rules. Database coverage pruning (Liu, et al ., 1998) for instance discard all rules that doesn't correctly cover at least one training case whereas Lazy pruning (Baralis et al., 2004) urging that pruning must be limited to those rules that will wrongly classify a trading case and keep all others. The former may discards some useful knowledge due to the sever pruning procedure while the latter often generates large size classifiers that need high computational load (Abu-mansour et al., 2010). In this thesis, we introduce five new rule pruning (PC, PC^{3,} FC, FPC and FPCC) in AC that construct accurate and moderated size classifiers. The proposed pruning methods are discussed in details in chapter 3.

1.5.4 New multiple rule prediction method

The ultimate goal of any classification system is to build a model (Classifier) from labelled training data, in order to classify unlabeled data objects known as testing data. Predicting the class labels of test cases by AC can be one of two types, either by predicting the class labels by the highest precedence single rule applicable to the test case (Single Accurate Rule Prediction) or prediction class labels by multiple rules (Group of Rules Prediction).However, the former suffers sometime from bias classification since using single rule prediction might favour one rule to predict most of the cases that satisfies its condition (discussed in further details in chapter 3). In this thesis, we introduce a new prediction method called dominant Class (DC) which based on group of rules. The proposed class assignment method is discussed in details in chapter 3.

1.5.5 New AC algorithm (MCAR2)

Recent experimental studies (Liu et al., 1998) (Thabtah et al., 2005) (Yoon et al., 2008) in data mining revealed that AC builds more accurate classification models with reference to accuracy than traditional classification approaches. The generation of large number of rules in AC make it hard for end-user to maintain and understand the classification models. We present a new Multi-Class classification based on association rule mining that significantly reduces the number of generated rules. The proposed algorithms employ new evaluation procedures that consider a rule as significant rule if its body partially covers the training case body.

Further, MCAR2 utilise group of rule in predicting a test case in order to avoid bias in classification. For multi-Class classification rules, we introduce An Enhanced Multi-class Classification based on Association Rules (MCAR2)".Chapter 4 discusses the proposed, model in details and shows the experimental results against a number of datasets from the UCI data repository. Experimental results show that MCAR2 outperforms popular AC and traditional classification techniques such as C5.0 and RIPPER RIPPER, C5.0\, CMAR, CPAR, MCAR and CBA.

1.5.6 Application of associative classification

In recent years, TC problem has attracted many researchers due to the availability of documents online, digital libraries and digital journals. TC involves assigning text documents to one or more pre-defined categories based on the content (Antonie M. and Zaïane O. 2003). Manual handling for TC is time and effort consuming. Despite the exponential growth of text documents, there are few AC research works on TC problem such as (Antonie M. and Zaïane O. 2004) (Chen et al., 2005) (Li et al., 2007).

Most of the research works on TC problem are using traditional machine learning approaches such as SVM (Vapnik, 1995), decision tree (Quinlan, 1993), and KNN (Yang, 1999) where a few attempts to tackle the problem of TC using AC including (El-halees, 2006) (Antonie and Zaïane, 2004) (Qian et al., 2005) (Chen et al., 2005).

In this thesis, the developed AC model has been applied to TC problem by adding a pre-processing step to transform the data into a suitable form for learning. Extensive experimental results against common English text benchmarks (*Reuter's mod-split*) using the developed AC algorithm (Discussed in chapter 4) revealed a competitive results when contrasting with other algorithms from A.

1.5.7 Experimental study in testing and comparing the developed pruning and prediction methods and the developed AC model with other approaches.

In this thesis, we have performed a large number of experiments comparing a number of classification approaches including, decision trees, rule induction and AC against different benchmarks including binary, multi-class and text benchmark problems along with discussions in order to highlight the strength points in the proposed methods. Particularly, we test the proposed pruning methods, perdition method and the developed AC model against wide range of UCI datasets and Reuter’s mod-split benchmark (Lewis .D, 1998). The criteria used in the comparisons are time taking in building the model, number of rules produced, classification accuracy, precision and recall and breakeven point BEP (Joachims, 1998) BEP is the point where precision equals recall.

1.6 General Structural Design

Two models were designed and used in this thesis, AC model and Associative Text Categorisation (ATS) model; both are shown in Figures 1.1 and 1.2.

1.6.1 AC model design

The AC model is working as follow: The user selects the training dataset (text file format) and then defines the required thresholds (*minimum support and minimum confidence*). The AC system starts processing the training data by producing the complete frequent *ruleitems* set and then produce the set of Class Association Rules. A subset of these rules is selected (interesting ones) through evaluation procedure. The selected rules are then used to form the classifier. Lastly, the classifier is applied on the testing dataset. In this thesis we developed MCAR2 algorithm which mines single label cases (each case is assigned to one class label only).

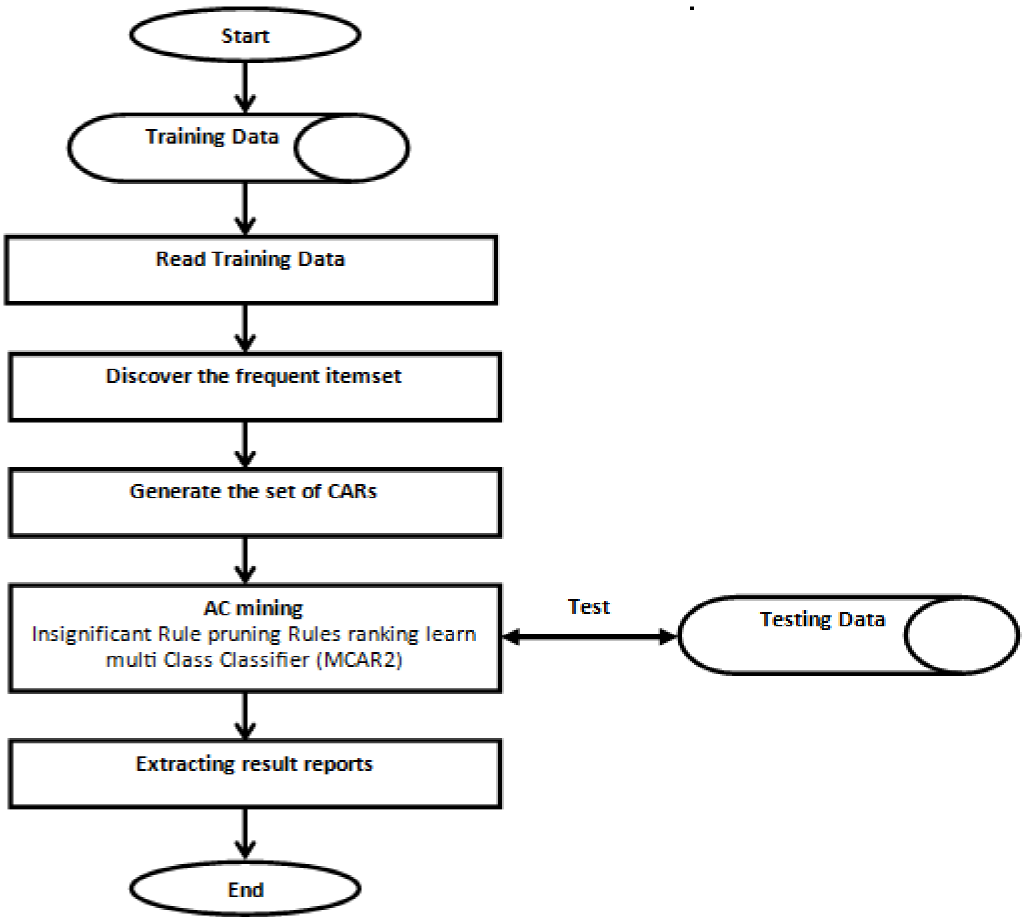
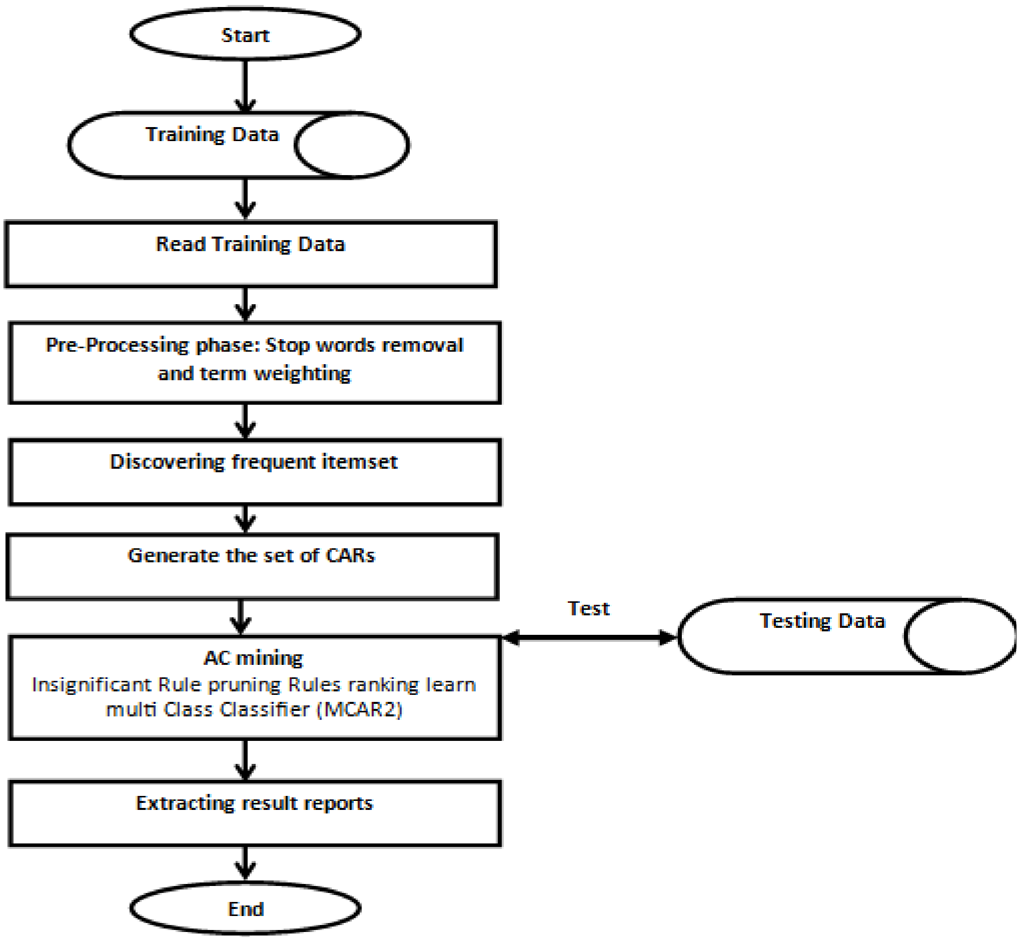

Start
Training Data
Read Training Data
Discover the frequent itemset
Generate the set of CARs
AC mining
Insignificant Rule pruning Rules ranking learn multi Class Classifier (MCAR2)
Test
Testing Data
Extracting result reports
End

Figure1.1: AC model design

1.6.2 Associative Text Classification model design

The proposed AC model is adapted to work on the TC problem. the model is working as follow; the user selects the training dataset (text file format), the dataset go through pre-processing phase that includes stop word elimination, terms extraction and weighting. The user defines the required thresholds (*minimum support and minimum confidence*), the AC system starts processing the training data by producing the complete frequent *ruleitems* set (terms), and then a set of Class Association Rules. A subset of those rule is selected (significant ones) to form the classifier. Lastly, the formed classifier is applied on the testing dataset (testing Documents which are pre-processed as well). In this thesis we adapt the developed MCAR2 algorithm which mines single label classifiers to deal with text data (Spars data).


Start
Training Data
Read Training Data
Pre-Processing phase: Stop words removal and term weighting
Discovering frequent itemset
Generate the set of CARs
AC mining
Insignificant Rule pruning Rules ranking learn multi Class Classifier (MCAR2)
Test
Testing Data
Extracting result reports
End
Figure 1.2: ATC model design

1.7 Thesis outline

This thesis consists of 6 chapters as follows:

Chapter One introduces the motivation, the research aims and objectives. The data mining and AC are briefly introduced; thesis contributions and the used structural models' design are demonstrated.

Chapter Two is literature review; the contents of the literature review focused on the association classification approach by discussing the common approaches in AC along with methods used at each step. Particularly, the common rule pruning and prediction approaches in AC have been discussed.

Chapter Three demonstrates the proposed pruning and prediction methods and highlights the Impact of rule pruning on the accuracy of the resulting classifier. Beside, experimental study in testing and comparing the developed pruning and prediction methods with other methods with respect to the classifier size, classification accuracy and model efficacy have been conducted.

Chapter four presents and discusses the proposed AC approach "Enhanced Multi-class Classification based on Association Rules" (MCAR**2)**. Experimental study in testing and comparing MCAR2 efficiency and effectiveness with other Algorithms has been conducted.

Chapter Five introduces text classification domain including, TC problem, TC phases and the common approaches used in each phase. Further, based on approaches and schemes proposed in previous chapters, Chapter 5 adapts MCAR2 to text classification problem. To the best of the author's knowledge there are very few attempts in adapting AC approaches to TC. Experimental study in testing and comparing the developed AC algorithm with other classification approaches from AC and Traditional classification approaches on TC has been conducted.

Lastly, Chapter Six summarises this thesis and contribution to knowledge. A discussion for the future work and directions is also described.

CHAPTER TWO

RULE PRUNING AND PREDICTION TECHNIQUES IN ASSOCIATIVE CLASSIFICATION

2.1 Introduction

It has been reported in many research works including (Liu, et al., 1998)(Thabtah et al., 2005)(Baralis, et al., 2008)(Hao et al., 2009)(Chen et al., 2012) that AC which builds rule based models lead to notable improvements on accuracy rate and effectiveness when contrasted with traditional classification data mining algorithms. Though, this approach suffers from two main problems

1) The large number of the generated rules which may be in tens of thousands or even hundreds of thousands when dealing with dense datasets. This often degrades the system efficiency and expands training time.

2) The biased classification and over-fitting during the classification phase (Prediction) which usually occurs due to relying on a single rule in predicting test data.

Normally, an AC algorithm operates in two phases, the rule generation and the classifier building. In the first phase, the set of frequent ruleitems are discovered and all rules satisfying the confidence threshold are then created. In the second phase, rules produced in the first phase are sorted according to certain parameters (Confidence, Support, etc) and then evaluated by pruning procedure(s) to discard redundant and misleading rules. Only the survived rules of phase one are used in the second phase to form the classification model.

The scope of this chapter is to review common rule pruning and class assignment techniques in AC mining and to show their impact on the classification accuracy. In other words, we intend to examine the impact of rule pruning and prediction steps on the classification accuracy and efficiency within AC.

There are different ways used by AC algorithms to evaluate the set of generated rules during the classifier construction step. For instance, the CBA algorithm (Liu, et al., 1998) utilize the database coverage pruning where rules correctly covering a number of training cases are marked as accurate rules and inserted into the classifier. However, the L^3 algorithm (Baralis, et al., 2004) claims that rules pruning like database coverage normally discards useful knowledge, therefore keeping all generated rules may give more power to the produced classifiers. In fact, their believe was supported by an evaluation result reported in (Baralis, et al., 2004) which employs a lazy heuristic that stores two kinds of rules: primary and spare in the classifier. The primary rules are those that cover certain training cases and the spare rules are those which do not cover any training case where a rule is added to the spar rules if a higher ranked rule covers correctly the selected rule training case(s). Spar rules stored in a compact-set aiming to use them during the prediction step especially when no primary rules cover a test case

There are other pruning techniques based on mathematics such as Chi-square(X^2) (Snedecor and Cochran, 1989) and pessimistic error (Quinlan, 1993). These techniques usually evaluate the rules statistically by measuring the correlation between rule body and the class it belongs to. Chi square X^2 tends to discard all rules that have negative correlation (according to a scoring function) between the rule body and the class label they contain. Pessimistic error is used in decision tree pruning such as C5.0 to evaluate the nodes in the decision tree by calculating its estimated error in order to decide whether to replace the node and its successors with a single leaf or leave it.

In the last step of an AC algorithm, assigning the right class for a test case becomes the key to success of the algorithm. There are many class assignment techniques that have been considered in the context of AC data mining, some of which relying on one rule to assign the class for the test case such as CBA(Liu, et al., 1998), and MCAR (Thabtah et al., 2005). In this approach, the first rule in the classifier applicable to the test case classifies it. On the other hand, some AC algorithms reply on group of rules such as CMAR (Li et al., 2001) and CPAR

(Yin X. and Han J. 2003) where all rules applicable to the test case predict the class label for a test case based on a certain scoring procedure.

In this chapter, we review in details common pruning and prediction techniques in AC and discuss the major issues related to AC. Finally, common AC algorithms such as CBA, MCAR, CMAR, etc are surveyed.

2.2 Pruning Techniques

A number of pruning techniques have been used in the context of data mining some of which is adopted from decision trees, others from statistics such as Pessimistic Error Estimation (Quinlan, 1993), Chi-Square testing (Snedecor and Cochran, 1989). These pruning techniques are employed either during rules discovery phase (Pre-Pruning) such as Pearson's correlation coefficient testing (Karl Pearson, 2003) or during the classifier construction phase (Post-pruning) such as Database coverage (Liu, et al., 1998) and Lazy (Baralis, et al., 2004). An early pruning step take place before generating the rules by eliminating those ruleitems didn't passed the *minsupp* threshold which may occur in the process of finding frequent ruleitems. This section discusses the current pruning techniques employed by the associative classification algorithms.

2.2.1 Database Coverage based Techniques

Rule Pruning Techniques that consider the rule as significant rule according to its coverage capacity are called database coverage techniques, following section list them and discusses their working mechanisms.

2.2.1.1 Database Coverage

The database coverage is the first heuristic that has been applied in CBA (Liu, et al., 1998) to select a rules subset to form the classifier. Database coverage method is a simple and effective pruning method; it evaluates the complete set of generated rules against the training dataset. Figure 2.1 shows the database coverage heuristic method in which starting with the highest ranked rule, all training cases that fully covered by the rule and the class is matched are marked for deletion from the training dataset and the rule gets inputted into the classifier. For a rule covering no training case (the rule body does not fully match any training case) then the rule is discarded. The database coverage method ends when either the training dataset gets is totally covered or there are no more rules to be added. In the case of no more rules are left for evaluation, the remaining uncovered training cases are used to generate the default class rule which represents the largest frequency class (majority class) in the remaining unclassified cases.

It should be noted that the default class rule is used during the prediction step in cases when there is no classifier rule applicable to the test case. Lastly, before the database coverage terminates, the first rule which has the least number of errors is identified as the cutoff rule. All the rules after this rule are not included in the final classifier since they often produce errors (Liu et al., 1998). Database coverage method has been criticized by (Baralis, et al., 2004) since in some cases it discard some useful knowledge. Alternatively, they urge that rich classifiers often provide useful and rich knowledge during the classification step.

```
Input: The set of sorted rules $R$ and the training dataset $D$
Output: The classifier $Cl$
 For each rule $r_i$ in $R$ do
     Mark all applicable cases in $D$ that  fully match $r_i$'s body
     If $r_i$ correctly classifies an case in $D$
  {       Insert $r_i$ into $Cl$
          Discard all cases in $D$ covered by $r_i$
 }
       If $r_i$ cover no cases in $D$
    {         Discard $r_i$
      }
 }
       If $D$ is not empty
 {          Generate a default rule for the largest frequency class in $D$
            Mark the least error rule in $R$ as a cutoff rule.
       }
```

Figure 2.1 Database coverage Pruning technique

2.2.1.2 Lazy Pruning

Lazy associative algorithms (Baralis, et al., 2002) (Baralis, et al., 2004) (Baralis et al. 2008) believed that pruning should be limited to the rules that incorrectly cover the training cases. Only these rules that lead to incorrect classification on the testing cases are discarded. Database coverage technique discards any rule that unable to fully cover a training case and class correctness. Alternatively, Lazy based method store all rules discarded by database like i

The lazy rule pruning invoked when the complete set of rules are discovered and ranked in descending order in which longer rules (rules with more items in its antecedent) are favored over general rules. For each rule starting from the highest ranked rule, if its cover correctly a training case, it will be inputted into the primary rule set, and all of its corresponding cases will be deleted from the training dataset. Whereas, if a higher ranked rule covers correctly training case(s) then it will be inserted into the secondary rule set (Spare rule-set). Lastly, if the selected rule is wrongly covers any training case, it will be removed. The process is repeated until all discovered rules are tested or the training dataset becomes empty. At that time, the output of this lazy pruning will be two rules sets, a primary set which holds all rules that cover correctly a training case, and a secondary set which contains rules that never been used during the pruning since some higher ranked rules have covered their training cases.

The distinguishing difference between the database coverage and lazy pruning is that the secondary rules set extracted by the lazy method are completely removed by the database coverage. In other words, the classifier resulting from database coverage pruning does not contain the secondary rules set of the lazy pruning, and thus it is often smaller in size than that of lazy based algorithms. This is indeed an advantage especially in applications that the end user can control and maintain the rules.

Empirical studies in (Baralis, et al., 2004), against a number of UCI datasets revealed that using lazy algorithms such as L^3 and L^3G sometimes achieved better accuracy rate than CBA like algorithms i.e. CBA2 and MCAR.. Lastly, we conclude that Lazy based algorithm are often scores high in term of effectiveness but low in the efficiency due to the large classifier size which takes more time in generating rules and learning the classifier.

2.2.2 Long Rules Pruning

A rule evaluation method that discards long rules (specific rules) that have confidence values larger than their subset (general rules) was introduced in (Li, et al., 2001). General rule with highest confidence value is used to prune the specific ones. In other words, it discards rules redundancy since many of the discovered rules have common shared attribute values in their antecedents that often results in redundant rules particularly when the classifier size becomes large.

The first algorithm used the long rules pruning was CMAR (Li et al., 2001). The set of rules firstly ranked according to the confidence, support and rule length, the rules are then stored in a CR- tree structure. A retrieval query over the tree is activated to check whether a rule can be removed. Chi square testing is applied on each rule R: $r_i \rightarrow c$, to determine whether r_i is positively correlated with c or not. The algorithm selects only rules that are positively correlated to form the classifier. There are a number of AC algorithms that employ this type of pruning including ARC-BC (Antonie and Zaïane, 2003), and Negative Rules (Antonie and Zaïane, 2004). Experimentation results reported in (Li et al., 2001) shows that using this pruning technique will positively affect the effectiveness when contrasted with other techniques.

2.2.3 Mathematical based Pruning

Some mathematical based pruning techniques have been proposed. Most of them tend to measure the correlation between different objects to decide whether they are correlated or not

2.2.3.1 Chi-Square Testing

The chi-square test (χ^2) proposed by (Snedecor and Cochran, 1989) is applied to decide whether there is a significant difference between the observed frequencies and the expected frequencies in one or more categories. It is defined as a known hypothesis that examines the relationship between two objects (Witten and Frank, 2000). The evaluation using χ^2 for a group of objects to test their independence or correlation is given as:

$\chi^2 = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i}$ Where:

e_i is the expected frequencies and o_i is the observed frequencies. When the expected frequencies and the observed frequencies are notably different; the hypothesis that they are correlated is rejected.

This method has been used in AC to prune negatively correlated rules. For example, a test can be done on every discovered rule, such as *r: x→c* , to find out whether the condition x is positively correlated with the class c. If the result of the test is larger than a particular constant, there is a strong indication that x and c of r are positively correlated, and therefore r will be stored as a candidate rule in the classifier. If the test result indicates negative correlation, r will not take any part in the later prediction and is discarded. The CMAR algorithm (Li, et al., 2001) adopts the chi-square 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

2.2.3.2 Pessimistic Error Estimation

Pessimistic error estimation was used in data mining within decision trees (Quinlan, 1993) which decides whether to replace a sub-tree with a leaf node or to keep the sub-tree . The method of replacing a sub-tree with a leaf is called "sub-tree replacement". The probability of an error at a node *v* is giving by the following relation that defines the pessimistic error:

$q(v) = \frac{N_v - N_{v,c} + 0.5}{N_v}$ Where

N_v Denotes the number of training cases that node v covers, $N_{v,c}$ is the number of training cases belonging to the largest frequency class at node *v*.

The error rate at sub-tree *ST*,

$$q(T) = \frac{\sum_{l \in leafs(T)} N_l - N_{l,c} + 0.5}{\sum_{l \in leafs(T)} N_l} .$$

The sub-tree *ST* is pruned if $q(v) \le q(ST)$.

Pessimistic error estimation technique has been exploited successfully in decision tree algorithms including C4.5 and C5.0 (Quinlan, 1998). In AC mining, the first algorithm which has employed pessimistic error pruning is CBA. For a rule *R*, CBA removes one of the attribute value in its antecedent to make a new rule *R'*, then it compares the estimated error of *R'* with that of *R*. If the expected error of *R'* is smaller than that of *R*, then the original rule *R* gets replaced with the new rule *R'*.

It should be mentioned here that CBA employs two pruning techniques, pessimistic error and database coverage. Some studies reported that employing sever pruning procedures may affect the accuracy rate (Baralis, et al., 2004) (Abumansour et al., 2010) (Thabtah et al., 2011).

2.2.3.3 Pearson's Correlation Coefficient Testing:

Pearson's correlation coefficient testing (Karl Pearson, 2003) is another statistical approach to measure the correlation between two objects. HMAC (Sangsuriyun, at.el.2010) is one of the associative classification approaches that uses this measure. After generating the set of CARs, HMAC uses two pruning procedures (1) Pearson's correlation coefficient procedure and (2) redundant rule, after the set of rules are being ranked according to the ranking procedure (Sangsuriyun, at.el.2010), HMAC starts with Pearson's Correlation Coefficient and apply it for every positive class rule RPC to measure the correlation strength between the rules antecedent i.e. the items and the consequent i.e. the class label. All rules with correlation measure π is below a predefined threshold will not be considered in the classifier. Three states could be found for a rule r_i, (1) the rule is said to be independent if the correlation measure between the antecedent and consequent $\pi = 0$, (2) is said to be positively correlated if $0 < \pi < 1$ and (3) negatively correlated if $\pi < 0$. Only the rules that are positively correlated are considered. The negative redundant rule pruning procedure is applied since some rules could share the same items in their antecedent.

Although it is revealed in their experimental results that algorithms using Pearson's test can result in gaining good accuracy results, it is difficult to validate this as insufficient experimental results are available and much of the information relating to their generation is absent.

2.2.4 Laplace Accuracy

Laplace accuracy (Clark and Boswell, 1991) is a post-pruning method, which is get invoked during the construction of a rule. It used to estimate the expected error ratio of a rule *r: $p_1 p_2 \ldots\ldots p_n \rightarrow c$*, the expected accuracy for a given rule *r* is calculated by the following formula:

$$\text{Laplace}\,(r) = \frac{(p_c(r)+1)}{(p_{tot}(r)+m)}$$ Where

$p_c(r)$ Indicates the number of training cases covered by *r* with class *c*, $p_{tot}(r)$ is the number of training cases matching *r*'s condition and *m* is the number of class labels in the domain. Laplace was adopted in associative classification by CPAR (Yin and Han; 2003). The method invoked after the rules being generated and sorted, the expected error is calculated for every rule in the set of potential ruleset, if the result above a predefined value the rule is then discarded which ensure that only those rules with best expected accuracy will be used in classification. One disadvantage of the algorithm is that (the rules are often with less quality than those generated by other AC algorithm, the reason is that CPAR is using greedy algorithm (FOIL) and rule *r* is generated for the remaining cases in training dataset instead of the whole dataset.

Experimental results reported in (Yin and Han; 2003) on the basis UCI repository showed that CPAR which uses Laplace accuracy algorithm showed promising results when compared with than CBA and CMAR.

2.2.5 Redundant Rule Pruning

In associative classification approaches, all of the attribute combinations are considered as a rule's body, thus rules used in building the classifier may share some training items and as a result, some specific rules might contain many general rules. The presence of the redundant rules in the classifier could be misleading as the number of the generated rules usually huge.

The method was proposed in (Li, et al., 2001) which discards specific rules that have a confidence value less than general rule. The method is working as follows: once the set of rules being generated and sorted, redundant rule evaluation is invoked to discard all rules where there are some general rules with higher rank and $I \subseteq I'$. This method notably reduces the size of the classifier.

Redundant rule pruning method have been used in several associative classification algorithms including CMAR (Li, et al., 2001), ARC-BC (Antonie and Zaïane, 2003), CACA (Tang and Liao, 2007) and contributed in enhancing their classification rate.

2.2.6 Conflicting Rules

In some datasets in which they considered dense datasets or multi-label i.e. multiple class labels associated with a training case, there is a possibility to have two rules with same antecedent associated with two different class labels, such as the following two rules: $x \rightarrow c_a$ and $x \rightarrow c_b$**,** conflicting rules pruning method (Antonie and Zaine, 2003) discards them and prevent them to take any role in the classifier. However, it has been reported in the experimental results by MMAC algorithm (Thabtah, et al., 2004) that such rules could represent useful knowledge.

To benefit from such rules, a recursive learning procedure have been developed by MMAC to combines what so called conflicting rules into one multi-label rule. For the above two rules, MMAC combines the two rules into the following multi-label rule: $x \rightarrow c_a \vee c_b$.

2.2.7 I-prune

Item prune is a recent proposed method in (Baralise and Garza, 2012). I-Prune is a pre-pruning method tends to mark uninteresting items based on interestingness measure (correlation measures e.g “Chi Square”, “Lift”, “Odd ration”) and remove them and use only interesting items to build a high quality rules which will be used in building the classification model. Consequently, such early pruning step will reduce the number of generated rule as well the time taken for learning the classifier.

Several AC algorithms such as CBA, CPAR, CMAR, and MCAR consider an item interesting according to the support count. Alternatively, I-prune selects only those are frequent and correlated to a class, Given an item *i* that is correlated to class *c*, an interestingness measure is given as follows: if *interestegness-measuer* (*i*,*c*)> *predefined-threshold* then *i* is selected else the item is discarded as soon as detected. Assume *I* is a subset of frequent and correlated items with respect to class *c*, only the rules that contains interesting items are generated. On the other hand, I-prune may inadvertently discard some item that might produce useful classification rules in later stages. Experimental results shows that Chi Square is the best correlation measure with respect to effectiveness (See (Baralise and Garza, 2012) for more details).

Lastly, I-Prune can be easily and effectively integrated with a number of Ac algorithms especially those are Apriority based algorithms such as CBA, CBA2 and MCAR.

2.2.9 PCBA based Pruning

Class imbalancing problem has not received big attention in the context of information retrieval and data mining. Classifiers with Imbalance class examples may increase the misclassification ratio (Liu et al., 2003) (Chen et al,. 2012).

To deal with class imbalancing in (Chen et al,. 2012), PCBA pruning method was proposed .Conventional AC algorithms used one fixed *minsupp* and *minconf* which might be working properly when dealing with balanced data but not for imbalanced one. Alternatively, the algorithm uses different *minsupp* and *minconf* values based on the rule distribution across classes.

PCBA adjusts CBA algorithm by proposing a new pruning method that filters the set of deemed CARs to be suitable to SBA algorithm which will lead to better accuracy in cases where classes are imbalanced. PCBA have improved two aspects in the original CBA (1) adopting sampling method which usually used to adjust the confidence of the minority classes that lead to enhancement on the ranking effectiveness and (2) decrease the level of class imbalancing that consequently makes minority classes more common. There are two sampling approaches, “under-sampling” and “over-samples”, due to the fact that over-sampling might cause over-fitting problems; PCBA adopts under-sampling approach in adjusting the confidence of the rare CARs. When under sampling used during the ranking procedure it controls the amount of positive rules by decreasing the amount of negative rules.

Second issue addressed in this work is setting multiple *minsupps* and *minconfs* thresholds. Setting single value is no**t** appropriate when dealing with imbalanced data since two problems might occur; when setting the support to high value, it becomes impossible to locate those rule from the minority classes on the other hand, setting the support to a low value in order to locate the rule from minority class will lead to a huge number of rules which need high computational cost and might degrade. Hence, PCBA employed different *minsupps* and *minconfs* values, the method adopted from (Liu et al., 2003) where the system users are required to define one *minsupp* called *t-minsupp* which changed according to the class distribution.

2.2.10 Discussions

AC approach which builds classification models by employing association rule mining often produce very large numbers of rules where some of them are significant and others are not. Hence, putting constraints on selecting the classification rules which is essential for a number of reasons: 1) the generated rules often have many of noisy, redundant and misleading rules that should be used in the classification step. 2) the time taking in building the model for classification would take much time and lead to increase the computational cost, 3) having some insignificant rules in the classifier may degrade the classification rate.

In the previous section, the most common rule pruning techniques used in the context of information retrieval and data mining have been reviewed, strengths and weaknesses were highlighted. Some of which is adopted from decision trees such as Pessimistic Error Estimation, others from statistics such as, Chi-Square testing and Pearson's correlation and AC such as database coverage and lazy pruning. These pruning techniques can be categorised based on the invoking period, Pre-Pruning (before learning the classifier) such as Pearson's correlation coefficient, I-prune, and Post-pruning (during the classifier construction phase) such as Database coverage, Lazy and others. Some of these algorithms favor the general rules (long rules) over specific believing that general rule implicitly contains the specific one and this will improve the model efficiency such as Database coverage and long rule pruning techniques, others have a different opinion, saying that those techniques favoring general rules loss some useful knowledge by discarding some good specific rules such as lazy pruning.

Generally, a rule pruning method either score well in reducing the time taking in learning and classification(model efficiency) and not for the classification rate (effectiveness) OR scoring well in the achieving competitive classification accuracy but time taken will increased, very rare evaluation techniques accomplished both aims.

Based on empirical studies and extensive experimental results and the investigations on the current pruning techniques we can infer that there should be a trade-off between the classifier size and the classification accuracy which can be accomplished by forming classifiers moderate sized. Next chapter demonstrates a number of pruning techniques that can generate moderated classifiers with competitive classification rate in a competitive exceptional time.

2.3 Prediction Techniques

The last step in the life cycle of any classification algorithm in data mining is to assign the appropriate class labels to test cases. This step is called class prediction or class assignment step. There is a number of different approaches for class assignment task in AC mining, some of which employs the highest ranked rule in the classifier, i.e. single rule prediction (Liu, et al., 1998) (Thabtah and Cowling, 2007) (Tang and Liao, 2007), and others uses multiple rules prediction , i.e. (Li, et al., 2001) (Yin and Han; 2003) (Thabtah, et al., 2011).

2.3.1 Single Rule Class Assignment Method(s)

The basic idea of the one rule prediction shown in (Figure 2.2) was introduced in CBA algorithm (Liu, et al., 1998). This method works as follows: the rules are sorted in descending order according to confidence and support thresholds, CBA iterates over the rules in the classifier and assigns the class associated with the highest sorted rule that matches the test case body to the test case. In cases there are no rules matches the test case body, CBA takes on the default class and assigns it to the test case.

```
Input: Classifier (R), test dataset (Ts), array Tr

Output: error rate Pe

Given a test data (T),
 1 ∀ test case ts in Ts Do
 2   ∀ rule r in the set of ranked rules R Do
 3     Find all applicable rules that match ts body and store them in Tr
 4      If Tr is not empty Do
 5          If there exists a rule r that fully matches ts condition
 6            assign r's class to ts
 7         end if
 8            else assign the default class to ts
 9              end
 10   empty Tr
 11    end
 12 end
 13 calculate the total number of errors of Ts;
```

Figure 2.2 single rule class assignment method

After the dissemination of CBA algorithm, a number of other AC algorithms have employed its one rule prediction method such as (Baralis and Torino, 2000), (Baralis, et al., 2002), (Thabtah, et al., 2005), (Tang and Liao, 2007), (Li et al., 2008), (Kundu et al., 2008,) (Niu et al., 2009).

2.3.2 Class assignment based on group of rules techniques

Single rule prediction method performs well especially when there is just a one rule applicable to test case *ts*. However, in cases when more than one rule with close confidence values is applicable to the test case, this method becomes questionable and selection of a single rule to make the class assignment is inappropriate. Using all rules contributes to the prediction decision in these cases is more appropriately. In the following section, the different multiple rules class assignment techniques are discussed.

2.3.2.1 Weighted Chi-Square Method

CMAR (Li et al., 2001) is the first AC algorithm that employed the statistical method weighted Chi-Square (*Max* χ^2) for class assignment task. CMAR class assignment method works as follows:

Given test case *ts*, and a set of the ranked rules *R*, the subset of rules, R_k that satisfies the test case condition is selected by the algorithm. If all rules in R_k have the identical class, then that class will be assigned to *ts*. If the rules in R_k associate with different classes, CMAR divides them into groups based on the classes and computes the strength of each group. The group's strength is identified by different parameters such as the support and correlation between the rules in a group. i.e. (*Max* χ^2). CMAR assigns the class with the largest group strength to the test case *ts*.

For a rule *R*: $Cond \rightarrow c$, assume *Support*(*Cond*) represents the number of training cases associated with rule body *Cond and Support*(*c*) denotes the number of training cases associated with class *c*. Also assume that $|T|$ represents the training dataset size. The *Max(* χ^2) of R_k is defined as:

$$Max\ \chi^2 = (\min\{Support(Cond), Support(c)\} - \frac{Support(Cond)Support(c)}{|T|})^2 |T| u$$

, where

$$u = \frac{1}{Support(Cond)Support(c)} + \frac{1}{Support(Cond)(|T| - Support(c))} +$$
$$\frac{1}{(|T| - Support(Cond))Support(c)} + \frac{1}{(|T| - Support(Cond))(|T| - (Support(c))}$$

Experimental results reported in (Li, et al., 2001) showed that classification procedures that employ a group of correlated rules for prediction slightly improve the classification rate when contrasted to other techniques.

2.3.2.2 Laplace based Method

CPAR algorithm is the first AC learning technique that used "Laplace Accuracy" to evaluate the rules and assign the class labels to the test cases during class assignment step. Once all rules are found, ranked and the classifier constructed, and a test case (*ts*) is about to be predicted, CPAR go over the rule set and marks all rules in the classifier that may cover *ts*. If more than one rule is applicable to *ts,* CPAR divides them into groups according to the classes, and calculates the average expected accuracy for each group. Finally, the class with the largest average expected accuracy value is assigned to *ts*.

Laplace Accuracy has been successfully used by CPAR algorithm (Yin and Han; 2003) to ensure that the largest rule(s) accuracy contribute in class assignment for test cases, which therefore could positively affect the classification accuracy. Fitcar (Cerf et al., 2008) is another AC algorithm that employed the prediction procedure of CPAR which is based on multiple rules. Empirical evaluation using different datasets from UCI repository revealed that algorithms that used Laplace such as CPAR algorithm achieves slightly higher classification accuracy than other algorithms such as CBA, decision trees and RIPPER.

2.3.2.3 Dominant Class and Highest Confidence Method(s)

Two closely similar prediction techniques that use multiple rules to predict the class labels for test cases were proposed in (Thabtah, et al., 2011). The first method is called "Partial Dominant Class", which marks all rules that are applicable (Partially match the test case body) to the test case, and then groups them according to class labels, and assigns the test case the class of the group which involve the largest number of rules applicable to that case. In cases where no rules are applicable to the testcase, the default class (Majority class) will be assigned to that case.

The second prediction method is called "Highest Group Confidence", which works similar to the "Partial Dominant Class" method in the way of marking and dividing the applicable rules into groups based on the classes. However, the "Highest Group Confidence" computes the average confidence value for each group and assigns the class of the highest average group confidence to the test case. In cases where no rule matches the test case, the default class will be assigned to that case.

2.3.3 Predictive Confidence

In class assignment step, the foremost weight considered in selecting the right rule for class assignment is the confidence value. However, (Do et al., 2005) argued that confidence which calculated from the training data is not enough alone to discriminate among rules. Thus, there should be other criterion for rule selection in prediction beside the confidence value .For instance, the "predictive confidence" measure that used to from the test dataset and for each rule in the classifier.

The predictive confidence is the average classification accuracy for a rule *r* when assigning classes to test data case. Consider for example a rule r_i: $ListOfItems \rightarrow c$, assume that there is "A" parameter represents the test cases that matching r_i condition and belonging to class label *c*, and "B" parameter represents the test cases matching only r_i condition. Now, when r_i is applied on the test dataset, r_i will correctly predict "A" test cases with prediction accuracy of

(A/B) which is simply the confidence value of (r_i) on the test dataset. This is predictive accuracy of the rule that has been implemented on a recent AC algorithm called AC-S (Do et al., 2005). This measure is employed to select the right rules for prediction. Experimental results reported in (Do et al., 2005) showed that AC-S algorithm is very competitive to common AC algorithms like CBA, and CMAR.

2.3.4 Discussion

There is a definite advantage of using just single rule in predicting class labels for test cases since only the highest applicable rule in the classifier has been used which is simple and scores high when it comes to the efficiency measurement. Further, the measure of choosing the rule for prediction represents a likelihood that a test case belongs to the appropriate class (Thabtah and Cowling, 2007). Nevertheless, utilizing just a single rule for class assignment has been criticized by (Li et al., 2001) (Liu et al., 2003) (Abumansour et al., 2011), that there could be multiple rules applicable to a test case with slightly different confidence values. Besides, for datasets that are unbalanced, using just one rule may be unsuccessful since there will be very large numbers of rules for the majority class (default class) and few numbers or no rules for the minority class. Thus, some scholars, e.g. (Vyas et al., 2008) (Yin and Han 2003) (Antonie and Zaïane, 2002) (Li et al., 2001) argue that making the decision based on a single rule leads to poor results and suggested using a group of rules for class assignment of test cases.

The main advantage of using multiple rules for prediction is that there is more than one rule contributing to the final decision, which greatly limits the chance of favoring a single rule to predict all test cases satisfying its condition. However, algorithms that rely on multiple rules in classifying test cases such as CMAR and CPAR do not consider the independence of the rules (Clark and Boswell, 1991) (Hu and Li, 2005) since during the training phase; cases are allowed to be covered by multiple rules with different class labels; this may cause rules dependency and conflicts. When a training case *t* is used to generate many rules during the rule discovery, then it is possible that in the prediction step, more than one rule with different class labels could be applicable to a test case similar to *t*. Further, there is rule dependency since once a rule classifies a training case during the rule evaluation phase; all other rules, which have used this case, are impacted. Algorithms that utilise one rule in the prediction step may sometimes produce good classifiers, but assume that the highest precedence rule is able to predict the majority of test cases satisfying its body.

2.4 AC algorithms

2.4.1 Classification based on Association (CBA)

CBA was proposed in (Liu, et al., 1998) as the first algorithm which used association rules to build classification systems. CBA starts by discovering the frequent ruleitems (<Attribute values>, Class) that survived the predefined *minsupp* threshold. Then, all ruleitems that pass the *minconf* threshold are converted into Class Association Rules (CARs) in the form of "if-Then" rules. CBA ranks the CARs according to rule's confidence, support and length and uses the database coverage pruning to discard insignificant rules which might lead to incorrect prediction decisions. Moreover, CBA employs the pessimistic error pruning to further removes negatively correlated rules.

Experimental results against a number of datasets for UCI reveled that CBA produced better accuracy than classic classification algorithms. However, when it comes to class assignment of test cases, CBA has been criticized as it is possible that there could be more than one rule applicable to a test case with similar confidence (Li et al ,. 2001) (Yin et al ,. 2003) (Chen et al., 2012).

2.4.2 Classification based on Multiple Association Rules (CMAR)

CMAR algorithm, proposed in (Li et al., 2001), use group of rules in predicting the class label of test data. CMAR was developed in attempt to overcome the bias prediction when relying on one rule. CMAR used "*Weighted Chi-square*" to measure the goodness of the rules under two criteria (1) the support and (2)the class distribution in the training data (see section 2.3.2.1).

To overcome the efficiency problem in training phase, CMAR adopted the efficient "FP-growth" algorithm (Han, et al., 2000) to find frequent ruleitems and generate the rules. An empirical study reported in (Li et al., 2001) showed that the effectiveness and efficiency of CMAR are better than that of CBA on a number of UCI datasets.

2.4.3 Classification based on Predictive Association Rules (CPAR)

CPAR proposed by (Yin X. and Han J. 2003) attempts to solve a number of main negative aspect in AC such as 1) reducing the high computational cost due to the large number of generated rules by adopting greedy method called "FOIL" (Quinlan and Jones, 1993) and 2) to deal with overfitting problem which occur due to employing the high-confidence rules in predicting class labels, by assigning the best *k* rules that satisfies the test case.

In rule generation step, CPAR proposed *"Predictive_Rule_Mining (PRM)"* method which modifies "FOIL". FOIL often generates a very small high quality rule-set where many discarded may represent useful knowledge. Alternatively, PRM generates larger set of predictive rules by keeping the covered training cases from the training dataset instead of discarding them but decreased their weight by multiplying a factor. For each item, CPAR compute its information gain after knowing the information that stored in *PNArray* which stores the following details, *P* & *N* values which indicates the number of negative and positive examples that satisfies the rule's condition and the number of negative and positive examples for each item *p* i.e. *P(p)N(p)* .

In classifying a test case, CPAR induce multiple rules for prediction and uses the best *k* best rules of each class in predicting class labels.

Experimental studies shows that CPAR improves the efficiency of the rule generation process when compared with popular techniques like CBA (Liu et al., 1998) and CMAR (Li et al., 2001). However, a number of falls for CPAR when evaluating the rules and classification have been reported in (Hao et al., 2009), classes with imbalanced distribution when may lead to favoring the example with more rules. When classifying test cases by CPAR, each class is treated evenly for examples which may lead to wrong classification.

2.4.4 Multi-class, Multi-label Associative Classification (MMAC)

A few studies were conducted on multi-label (A case may belong to more than one class) classification if contrasted with multi-class (more than two classes in the dataset) classification; The MMAC algorithm (Thabtah et al., 2004) considers the problem of multi-label data in AC. It comprises three steps: rules generation, recursive learning to learn the classifier and class assignment. As a first step, MMAC scans the training dataset once to discover the frequent ruleitems and generates the complete set of CARs from these items through employing fast intersections method called Tid-list (Zaki and Gouda, 2003). In the second step, MMAC proceeds to discover rules that pass the *minsupp* and *minconf* thresholds from the remaining cases in the training dataset until no further frequent ruleitems can be found.

The set of candidate rules (CARs) are generated and ranked according to their confidence, support, cardinality and class distribution frequency. In MMAC, The class labels are ranked based on their distribution, for example a rule r associated with two labels l_1 and l_2, l_1 precedes l_2 if it has a higher presentation in the training dataset, Hence, the rule r is represented in the following form: $r \rightarrow l_1 \vee l_2$. The set of rules is tested against test data after being evaluated on the training dataset by using Database coverage method (Discussed in 2.2.1.1).

The class assignment procedure in MMAC is simple, In classifying a test case, starting with the first rule in the classifier, the first rule satisfies a test case classify it. Experimental results on 28 different datasets (Merz and Murphy, 1996) showed that MMAC algorithm is an accurate and effective classification technique, highly competitive and scalable in comparison with other traditional and AC approaches such as PART, RIPPER, and CBA.

2.4.5 Multi-class Classification based on Association rule (MCAR)

MCAR algorithm which proposed in (Thabtah et al., 2005) consists of two steps, Rule generation and classifier construction uses an efficient approach for discovering frequent ruleitems. MCAR employs a rule ranking method to ensure detailed rules with high confidence are kept for classification. In the first phase, MCAR pass over the training dataset once to discover frequent 1-*ruleitems*, and then combines *1-ruleitems* to produce candidate *ruleitems* involving more attributes i.e. of size two and three and so forth.

MCAR finds frequent *ruleitems* of size *k* by appending disjoint frequent itemsets of size *k*-1 and intersecting their rowIds. The result of a simple intersection between rowIds of two item sets gives a new set which holds the rowIds where both itemsets occur together in the training data. This set along with the class array, which holds the class labels frequencies and was created during the first scan, can be used to compute the support and confidence of the new *ruleitem* resulted.

MCAR adds upon previous rule ranking approaches a new criterion that looks at the class distribution frequencies in the training data and prefers rules that are associated with the class with more frequency, e.g. If two rules, r_1 and r_2, have the same confidence, support and size, but r_2 is associated with a class that occurred more frequently in the training data than that of r_1, MCAR selects r_1 on r_2 during the ranking step. In cases all criteria the same, then MCAR selects one randomly. After generating the set of candidate rules, MCAR invokes the rule evaluation procedure which was adopted form (Liu, et al., 1998).. Lastly, in classifying a test-case, MCAR starts with the first rule in the set of ranked rules, the first rule applicable to a test-case classify it. In case no rule from the set of ranked rules is applicable to the test-case, the default (Majority) class will be assigned to the test case.

Performance studies on a number of datasets from UCI data collection indicated that classifier produced by MCAR is highly competitive when compared with traditional classification algorithms such as RIPPER and C4.5 in term of prediction accuracy. Furthermore, MCAR scales well if compared with popular AC approaches like CBA (Liu et al., 1998) with regards to prediction power, rules features and efficiency. On the other hand, we believe that MCAR can perform well if it would reconsider the pruning method; the current method used by MCAR losses representative knowledge in some cases due to the sever pruning procedure used in the evaluation step (Baralis, et al., 2002). Furthermore, several research works including (Vyas et al., 2008) (Yin and Han 2003) (Antonie and Zaïane, 2002) (Li et al., 2001) provide evidences through empirical studies that making the decision based on a single rule (such as the one adopted in MCAR) leads to poor results and suggested using a group of rules for class assignment of test cases.

2.4.6 Class based Associative Classification CACA

CACA proposed in (Tang and Liao, 2007) adopts vertical data representation for frequent item discovery to enhance classification efficiency and effectiveness. CACA works as follows:

Given a training dataset *T* with *M* classes, CACA starts by dividing the attribute values for all classes into smaller *k* ones for each class in to narrow the searching space. All frequent itemsets are transformed into vertical data representation for generating the rules and learning the classifier simultaneously, CACA stores and ranks the generated rules in an Ordered Rule Tree that of CR-Tree in (Li et al., 2001). For each rule generated, if it satisfies the predefined thresholds, its attributes values will be stored as nodes in the CR-Tree whereas the last node in the leaf stores the rule's information such as class labels, support and confidence. Nodes placed closer to the head are higher priority than those a bit below. Once a rule is generated and stored in the CR-tree in a ranked order, CACA checks whether this rule is redundant or not, if so then the rule will be discarding. In classifying a testcase, the first matched rule is used.

Experimental results suggested that CACA performs better in term of accuracy and computation time than MCAR (Thabtah et al., 2005) on a number of datasets from UCI.

2.4.7 ACCF Associative Classification based on Closed Frequent Itemsets

Li et al. , 2008) proposed a new method, called ACCF. An Itemset "X" is a closed frequent Itemset in a data set S if no proper super-itemset Y such that Y has the same support count as X in S, and X satisfies minsupp. This method extends an efficient closed frequent pattern mining method called Charm to discover all frequent closed itemsets (CFIs) and their tid sets (Zaki and Hsiao C-J, 1999). This would help in the generation of the CARs. The experiments on 18 data sets from UCI repository (Merz and Murphy, 1996) showed that ACCF is consistent, highly effective at classification of various kinds of data sets and has better average classification accuracy in comparison with CBA.

2.4.8 Lazy based associative classification

Eager associative algorithms extract the set of CARs form the training data while a Lazy AC algorithms (*L^3*) proposed in (Baralis, et al., 2008) induce specific rules for each test case by projecting the training data only for features then the set of CARs is induced, ranked and only the best ones are used. lazy pruning method used in *L^3* discards the harmful rule that leads to wrong classification yield two sets of rules when invoking the classifier for prediction , one is the set that contains the high quality rules which considered for classification (those are correctly cover training cases), if all rules in this set failed to classify a test case then the second set of rules that contains rule usually discarded by other AC algorithms is used, Several versions of lazy based approaches have been developed (Baralis, et al., 2002, 2004, 2006) each of them was fixing some deficiencies in its predecessor. (Baralis, et al., 2008) is the recent version that employs compact form for representing rules which will be used later for generating the set of rules.

L^3 Learn the classifier through two steps (1. Classification rules mining that exploits the compact representation to perform rule mining with very low *Minsupp* value, the process of mining the rules yield a high quality selection to be used in the evaluation step. (2. Classification rule evaluation. *L^3*classifiers believe in the rule rich classifiers because they often provide useful and rich knowledge during the classification step. Hence, *L^3* approach reduces the pruning amount by using confidence, chi-square to discard neither representative nor correlated rules.

L^3 classifier has two levels of CARs, Level I that contains rules that correctly cover training cases during the learning step and level II that stores the compact rules. The classifier starts with first rule in level I set, if a rule matches a *ts* it classify it. In case no applicable rules in level I then level II is considered.

Experimental evaluation using real and synthetic datasets shows that *L^3* slightly improves the classification accuracy when contrasted with other existing AC Algorithms. However, still *L^3*

spent more time than other AC algorithms during the learning and classification steps due to the rich learning complexity.

2.4.9 Hierarchical Multi-Label AC using Negative rules HMAC

Hierarchical Multi-Label AC algorithm was developed in (Sangsuriyun, at.el.2010). HMAC uses negative rules in predicting the class labels for test cases. "mutli-label rule" denotes a rule that have more than one class in its consequent such as r:I $\rightarrow c1, c2$, "negative association rule" (Gan et al., 2005). Given a rule R: Ra is the rule antecedent which is a combination of three items, X: positive items, N: negative item and NP: negation of positive items and Rc is the rule consequent which can be one of three types too, PC: positive class, NC: negative class or NPC: negation of the positive class means none simultaneous presence of classes; a negative rule can be given as$Pa \rightarrow Pc$.

After generating the complete set of rules (CARs), HMAC invokes the ranking procedure based on a number of parameters, *F-measure*, *Jaccard*, *Support*, *ActOcc* and *rule length*. HMAC replaced the confidence parameter by *F-measure*, *Jaccard*. In evaluation the set of rules, the algorithm then fires two pruning procedures (1) Pearson's correlation coefficient procedure (Section 2.2.3.3) and (2) redundant rule pruning (Section 2.2.5). The classification uses only those rules that are not redundant and are positively correlated. Lastly, in classifying a test case *t*s, its compared with the set of ordered rules, if *ts* matches a rule positive class without any rules negative class it classify it, if failed to match any rule then HMAC moves to the next rule set, if still didn't match any rule in all sets then the default class is assigned. Although it has been reported in this article that algorithms using Pearson's test can result in good accuracy but it is difficult to validate this as inadequate experimental results are available and much of the information relating to their generation is absent.

2.4.10 Probabilistic CBA

Very rare since very little attention was paid to Class imbalancing, SBA (Liu et al., 2003) introduced a scoring mechanism for training cases in order to reveal the *likehood* to that the case belongs to rare class. Although SBA employs pessimistic error estimation measure to perform pruning, but still the number of rules is large and resulting in complication upon scoring procedure. New AC algorithm called PCBA proposed a new pruning method (Chen et al., 2012) aiming to improve CBA accuracy of rare data cases. As discussed in section 2.4.1, CBA deal only with Class association rules of the form $I \rightarrow \boldsymbol{c}$, where I is the set of items and c is the class. These CARs are then ranked according confidence, support and rule generated first. It should be noted here that CBA ranking procedure is adequate only when classes are evenly/ semi even distributed (Balanced data). On the other hand imbalanced data when tested often degrade the classification accuracy.

CBA produce Classifier such that $C=(r_1, r_2, r_3,.....r_i, Defult\_class)$ where the first class applicable to the test cases classify it. In cases where no rules are applicable CBA assign the default class to the case. Alternatively, SBA is introduced probabilistic classifier that assigns score for each case: $\boldsymbol{\mathcal{F}}: \boldsymbol{di} \rightarrow [\mathbf{1}, \mathbf{0}]$ where each di is mapped to a real value$(\boldsymbol{di})$. The Following equation $\boldsymbol{T} = \{\mathbf{di} | \boldsymbol{\mathcal{F}}(\mathbf{di}) \geq \mathbf{t}\}$ denoted that each pair of (*P*. Classifier and threshold *t)* defines a binary classifier. Figure 2.4 explain the PCBA algorithm steps.

```
Input:  t-minsupp, dataset D.
Output: Set of CARs that form C.

1.  $D_{new}$=Under-sampling ($D$),
2.  Compute minsupp (p), minconf (p), minsupp (n) and minconf (n)
3.  $R$, conf (under-sampling )= $CBARG$(D, minsupp (p), minconf (p), minsupp (n) and minconf (n)),
4.  Sort R based on Conf (under-sampling)
5.    For each $r$ in $R$ do
6.      Temp= $\theta$
7.          For each $d$ in $D$ do
8.             If $d$ applicable to $r$ condition then retrieve $d$.id in tempt and mark $r$.
9.  }
10.                If $r$ is marked then
11.   insert $r$ into $C$.
12.   Remove all cases those in tempt from D.
13. }
14.  }
```

Figure 2.3: PCBA algorithm

Evaluation on six imbalanced dataset (benchmarking and real life applications) is conducted; results revealed that the proposed algorithm performs better than C5.0 and SBA.

2.5 Summary

In this chapter, different AC algorithms have been reviewed as summarized in Table 2.1 Different approaches used in each algorithm including rule discovery, rule ranking, rules evaluation, and class assignment. The effectiveness for their model, the issue of efficiency has been discussed too. Research studies on AC to date have been focused on the general classifications problems including the exponential grows of the rules generated by AC algorithms, bias classification when assigning classes to the test cases. All AC algorithms aiming to construct an effective classifier that overcomes the above mentioned issues.

In AC context, many algorithms have been successfully adopted to construct effective classifiers such as CBA, CMAR, CPAR, MCAR, HMAC, where only the accurate, positive and significant rules are kept for classification. Sometime the discarded rules may represent a useful knowledge which domain users can make use of them (Baralis et at., 2008). For the highly correlated datasets, there should be additional ranking criteria beside the confidence, support, length and rule generated first in order to discriminates rule and reduce or eliminate the random selection.

Rule evaluation (Pruning) procedures are used to reduce the size of rules and minimize the possibility of over-fitting. Further, several AC algorithms have adopted horizontal data representation, Apriori for instance require multiple passes over the database in order discover the frequent ruleitems which may degrade the system efficiency and necessitate high computational cost. Alternatively, other AC algorithms have adopted vertical data representation which requires only one pass over the database to find the set of single frequent items, and then frequent *ruleitems* of size *k* by appending disjoint frequent itemsets of size *k*-1 and intersecting their rowIds. This indeed reduces the CPU time.

Table 2.1: summary of AC algorithms

Algorithm	Pruning Method	Prediction method	Reference
CBA	- Pessimistic error. - Database coverage	Maximum likelihood	(Liu et al., 1998)
CMAR	- Chi-square (X^2). - Database Coverage. - Redundant rule	Multi-Label- CMAR	(Li et al., 2001)
CPAR	- Laplace Accuracy	Multi-Label- CPAR	(Yin & Han, 2003)
MMAC	- Database coverage	Maximum likelihood	(Thabtah et al., 2004)
MCAR	- Database coverage	Maximum likelihood	(Thabtah et al., 2005)
CACA	- Compact set. - Redundant rule	Maximum likelihood	(Tang and Liao, 2007)
ACCF	- Pessimistic error. - database coverage	Maximum likelihood	(Li X. et al. , 2008)
Lazy Based AC	- Lazy pruning	Lazy Maximum likelihood	(Baralis, et al., 2008)
ICPAR	- Laplace Accuracy	Multi-Label- ICPAR	(Hao et al., 2009)
HMAC	- Pearson's Correlation Coefficient testing - redundant rule pruning	HMAC positive rules - Maximum likelihood	(Sangsuriyun, at el.,2010)
PCBA	- PCBA pruning	SPA probabilistic	(W-C et al., 2012)

CHAPTER THREE

THE PROPOSED RULE PRUNING AND CLASS ASSIGNMENT APPROACHES

3.1 Introduction

AC is an approach which integrates association rule mining and classification to improve the performance of traditional classification algorithms in regards to predictive accuracy. A number of studies including (Li et al., 2001) (Yin and Han, 2003) (Thabtah, et al., 2004) (Thabtah, et al., 2005) (Li et al., 2008) (Niu et al., 2009) revealed that AC approach is able to derive more accurate classifiers than traditional classification techniques such as decision trees (Quinlan, 1993), and rule induction (Cohen, 1995). However, AC approach suffers problems such as the exponential growth of rules since in association rule mining all the correlations among the items and the class are discovered as rules. The large number of rules makes humans unable to understand or maintain. Hence, cutting down the number of those rules by imposing pruning procedures to discard redundant, noisy and misleading rules and keep the significant ones certainly becomes very important task.

There have been many attempts intending to minimize the size of classifiers formed by AC approaches, mainly focused on not allowing rules either misleading , redundant noisy from participating in forecasting test cases in the last step in an AC algorithm. Discarding such rules can make the classification process more accurate.

Another common problem associated with the class assignments of test cases. Class assignment task in AC algorithms is performed often by a single dominant rule. This rule is usually the highest sorted rule for the test case (The rule body match the test case attribute values). However, there could be multiple matched rules in the classifier contained within the test case which makes applying one rule in prediction highly undesirable.

This chapter is aiming to investigate the rule pruning and class assignment steps in AC mining in order to

a) Eliminate unnecessary rules in the evaluation sub-step against the training dataset when constructing the classifier.
b) Enhance the class assignment (accuracy rate) by utilizing group of rules procedure that in the class assignment decision and therefore improving the classification accuracy.

Particularly, and for pruning step, we develop different rule pruning procedures and implement them within a new developed AC algorithm called MCAR2 (See Chapter 4). These pruning procedures are tested against a collection of datasets from the UCI repository (Merz and Murphy) and then compared with other existing pruning procedures in AC such as the database coverage and Lazy pruning. The ground bases of the comparisons are the number of rules derived and the classification rate. The proposed pruning procedures use the concepts of full and partial match for considering the rule significant during the process of constructing the classifier:

DEFINITION 3.1: We say that a rule *r* partial covers training case *t* if *r* contains at least an item in its antecedent that exists in *t*.

DEFINITION 3.2: A rule *r* fully covers training case *t* if all *R* items are in *ts*.

For class assignment, we propose new method that relies on a group of rules to make prediction decision. This enables the multiple rules in assigning the right class for the test case which consequently may enhance the prediction rate. Experimental results in Section 3.5 using datasets from UCI repository reveal that using group of rules prediction methods such as the ones proposed in section 3.4 generate more accurate and powerful classifiers if contrasted with single rule class assignment methods such as those employed by CBA, and MCAR AC algorithms.

3.2 The Proposed Pruning Approaches

As mentioned earlier in the previous section, the proposed pruning procedures aims to (1) Cut-down the number of rules, (2) Examine the impact of pruning on classification accuracy. Two criteria were considered when evaluating a rule during the classifier construction. First, the correctness of the rule's class with that of the training case, and second the type of matching between the rule body and the test case attribute values. The second criterion means whether the evaluated rule body is contained within the test case fully or partially. We have considered both criteria in different rule pruning procedures in order to come out with the procedure that has the least negative effect on classifiers.

Partial matching occurs when one of the attribute values of a rule matches at least one item of a training case *T*. Some other proposed pruning procedures consider that beside the partial matching the correctness of the class *R* i.e. the class of the rule must match that of the training case and that of *T* class such "Partial Coverage" (PC^3) that discussed in Section 2.2.1.1. Other pruning procedure does not require the correctness of *R* class and that of *T* class to overcome the problem of overfitting the training dataset such as "Partial Coverage Correctly Classify" (PC). We have investigated all cases of partial matching, and full matching in both scenarios:

1) The necessity of class correctness between the rule class label and the training case class label. And 2) And without class correctness

The main reason for considering partial matching when forming the classifier is to give a chance for more rules to participate in the class label assignment of test cases which accordingly may enhance the accuracy rate since more than one rule is used in the prediction. In this section, we show the impact of considering partial matching and class correctness on the classification accuracy rate. In this regard, there are several research questions in which this chapter attempts to answer:

- When a rule body is fully matching a training case, are the accuracy rate and the classifier size affected? Here we investigate two scenarios; with class correctness and without checking the class correctness.
- When a rule body is partially matching a training case during the classifier forming, are the accuracy rate and the classifier size affected? Here we investigate two

scenarios (class correctness, without checking the class)

- Is it essential that the training case class must match that of candidate rule class in order to consider that rule when forming the classifier?

In this chapter, we propose five new rule pruning procedures two of them employ partial matching, one employ full matching and two use both criteria (Hybrid). Partial coverage pruning procedure (PC) considers partial matching between the evaluated rule and the training case without requiring class correctness between the rule class and that of training case (Section 3.2.1.1). On the other hand, Partial covering with class correctness (PC^3) considers partial matching but requires class match between the rule and the training case's class to consider the rule as potential rule (Section 3.2.1.2). For full matching procedures, “Full Coverage” (FC) necessitates that a rule body is fully contained in the training case without class correctness requirement (Section 3.2.1.3). “Full & Partial Coverage” (FPC) is a hybrid method considers a rule as significant if it is full matching a training case body, if failed to cover any from the training data then it checks the rule applicability based on partial matching without class correctness criterion (Section 3.2.2.1). Finally, “Full & Partial Coverage Correctly Classify” (FPC^3) is similar to (FPC) pruning procedure but it necessitates class correctness (Section 3.2.2.2).

In this section, we present the proposed pruning methods along with an example to explain each method. Table 3.2 shows an example of rule based model consists of fifteen candidate rules that were produced by an AC algorithm called CBA (Liu, et al., 1998) using *minsupp* and *minconf* of 2% and 40%, respectively, from the training dataset shown in Table 3.1.

Table 3.1 Example data from weather dataset

	Outlook	Temperature	Humidity	Windy	Play/Class
1	sunny	hot	high	false	no
2	sunny	hot	high	true	no
3	overcast	hot	high	false	yes
4	rainy	mild	high	false	yes
5	rainy	cool	normal	false	yes
6	rainy	cool	normal	true	no
7	overcast	cool	normal	true	yes
8	sunny	mild	high	false	no
9	sunny	cool	normal	false	yes
10	rainy	mild	normal	false	yes
11	sunny	mild	normal	true	yes
12	overcast	mild	high	true	yes
13	overcast	hot	normal	false	yes
14	rainy	mild	high	true	no

Table 3.2 Example of a rule-based model (potential rules) from weather dataset

RuleID	RuleDesc	Rule Support	Rule Confidence	Rule Rank
1	True→yes	0.214286	0.5	13
2	High ^mild→yes	0.142858	0.5	14
3	False^high→yes	0.142858	0.5	15
4	High→no	0.285715	0.5714	12
5	high^true→no	0.142858	0.6667	10
6	high^hot→no	0.142858	0.6667	11
7	false^hot→yes	0.142858	0.6667	9
8	False→yes	0.428572	0.75	6
9	Cool→yes	0.214286	0.75	7
10	Cool^normal→yes	0.214286	0.75	8
11	Normal→yes	0.428576	0.8571	5
12	Overcast→yes	0.285715	1	1
13	normal^sunny→yes	0.142858	1	3
14	hot^sunny→no	0.142858	1	4
15	high^sunny→no	0.214286	1	2

Before pruning starts, all candidate rules are ranked in descending order as per the following rule ranking procedure:

Given two rules: r_1 and r_2,

(1) $r_1 > r_2$, if r_1 has a higher confidence than r_2.

(2) If confidence is similar for both rules, then $r_1 > r_2$ if it has a higher support.

(3) If the confidence and support values are the same, then $r_1 > r_2$ if it has fewer numbers of items in its antecedent than that of r_2.

When we apply the above rule sorting method of the rules within Table 3.2, Rule IDs 12-15 have the same confidence but Rule ID-12 is ranked higher due to its larger support than Rule ID-15. For rules 13-14, they have the same confidence and support values and we must select one randomly in this case since both rules have similar length, confidence and support values. In chapter 4 section (4.2.3) we add another new criterion called rule class frequency in the training data set to distinguish further between rules. In this case, Rule ID-13 has a higher rank than that of Rule ID-14 since it is associated with class "Yes" in Table 3.2 which has larger frequency than class "No" (In this example class "Yes" frequency is 9 and it is larger that of "No" which is only 5).

The proposed rule pruning methods group the rules into two main groups, one on Single criterion Pruning and one based on two criteria; two criteria Pruning.

3.2.1 Single Criteria Pruning

We present three pruning procedures in which they use single criteria, these are, PC, Partial PC[3] and FC. Discussions on each procedure are given in the following sub-sections. Following section demonstrate the proposed pruning methods:

3.2.1.1 Partial Coverage (PC)

The basic idea of PC pruning (shown in Figure 3.1), is that a rule *r* is considered (added to the classifier) if at least one item of its body is contained in the training case regardless the class correctness. Consider the rules in Table 3.1, r_{12} (Overcast$\rightarrow$yes) is the highest ranked rule. When applying that rule on the training cases (Table 3.2) in the PC pruning, we find that it covers cases 3, 7, 12, 13, so r_{12} is added into the classifier and cases 3, 7, 12, 13 are removed from the training set. We proceed with the second ranked rule r_{15}, we found it's applicable to seven remaining cases in the training data set (1, 2, 4, 8, 9, 11, 14), we add this rule into the classifier and remove the covered cases.

We proceed to the next ranked rule(s) (r_{13}, r_{14}) and we select r_{13} since it is associated with larger frequency class than r_{14}. All training cases covered by this rule, e.g. (Cases 5,6,10) are deleted and the rule gets inputted into the classifier. The training data set becomes empty once *r_{13}* got inserted and we output the classifier. Eventually, the resulted classifier using this pruning method will contain three rules, (*r_{12}*, *r_{15}* and *r_{13}*) since these rules cover all the training cases. We stop learning when either all rules are used or all training cases got covered. Here, we stopped at the third round as all training cases were covered.

```
Input: Training data set T and Set of Ranked Rules R
Output:  Classifier h
1  R' = sort(R);
2  ∀rule ri in R'
3     Find all applicable training cases in T that match ri's condition Where
      at least one item of ri's condition in ti
4        Insert the rule at the end of h
5           Remove all training cases in T covered by ri.
6  else
7           Discard ri and remove it from R
8   end
9  Next r
```

Figure 3.1: PC pruning method

3.2.1.2 Partial Coverage Correctly Classify (PC^3)

The basic idea of the PC^3 pruning (shown in Figure 3.2) is that a rule *r* is added to the classifier if at least one item of its body is found in the training case (partially match) and that the class of the rule is identical the training case class. Consider Table 3.2 and starting from the top ranked rule (*r_{12}*), when applying this rule on the training data set of Table 3.2 and according to PC^3, we find that it is applicable to cases (3, 7, 12, 13) and with similar class labels "Yes". So *r_{12}* gets inputted into the classifier and all of its covered cases are removed from the training set. We proceed to the second sorted rule ID (*r_{15}*), we found it covers cases 1, 2, 8 and 14 so the rule gets inserted into the classifier and its covered cases are discarded. We continue to the next rule ID (*r_{13}*) and apply it on the training data set and remove its training cases (5,9,10,11). The next rule(s) IDs (r_{14}, *r_{11}*, *r_{10}*, *r_9*) have no training cases coverage so they will be discarded. The next rule ID (*r_8*) (False→yes) covers a single training case 4 so it gets inputted into the classifier and the next two rules IDs ($r_{6,}$ r_7) have no coverage so they are discarded. The last rule that to be considered by PC[3] is rule ID (r_5) and at that time the pruning procedure terminates and only Rule IDs ($r_{12,}$ $r_{15,}$ $r_{8,}$ r_5) are the classifier rules and all other rules are discarded.

We stop learning when either all rules are tested or all training cases got covered. Here, we stopped when all training cases were covered and four rules ($r_{12,}$ $r_{15,}$ $r_{8,}$ r_5) have not even been tested on the training data set since their training cases are covered by higher rankled rules ($r_{1,}$ $r_{2,}$ $r_{3,}$ r_4). It should be noted that some rules have been deleted since they were unable to cover any training data case even when they have been tested, i.e. ($r_{14,}$ $r_{11,}$ $r_{10,}$ $r_{9,}$ $r_{3,}$ r_4).

```
Input: Training data set T and Set of Ranked Rules R
Output:  Classifier h
1  R' = sort(R);
2  ∀rule ri in R'
3     Find all applicable training cases in T that match ri's condition Where
      at least one item of ri's condition in ti and ri is correctly classify t
4         Insert the rule at the end of h
5            Remove all training cases in T covered by ri.
6  else
7                        Discard ri and remove it from R
8    end
9  Next r
```

Figure 3.2: PC^3 pruning method

3.2.1.3 Full Coverage (FC)

This pruning procedure is listed in Figure 3.3 in which a rule *r* is considered significant rule if its body (attribute values) all within the training case (full matching) regardless the class correctness of the rule with that of the training case. Let's revisit Table 3.1 and apply the FC pruning on it to generate the classifier. Starting with the top sorted rule (r_{12}), the training cases that are associated with this rule are 3, 7, 12, 13 so r_{12} is marked significant and its associated training cases are removed. The process continuous and rules (r_{15}, r_{13}) has database representation and precisely cover training cases (1, 2, 8) (9,11) respectively. These two rules get marked as significant rules and their training cases are deleted from the training dataset. The next rule (r_{14}) has no data coverage so it will be discarded and the next rule ID (r_{11}) has two data coverage in cases (5, 6, 10) so it will be significant rule and its cases get removed. The following two rules IDs (10, 9) have no data coverage and they are deleted and the rule ID (r_8) has one data coverage (4) so it is marked as a significant rule and its covered training case is discarded. The last rule that has data coverage is rule ID (r_5) which has one data coverage (case 14). All remaining rules are discarded since the training data become empty. This pruning procedure produces 5 significant rules classifier ($r_{12,} r_{15,} r_{13,} r_{14}$, r_5).

We stop learning when all rules are used or all training cases got covered. Here, we stopped after evaluating eleven rules.

```
Input: Training data set T and Set of Ranked Rules R
Output:  Classifier h
1   R' = sort(R);
2   ∀rule ri in R'
3      Find all applicable training cases in T that fully match ri's condition
4         Insert the rule at the end of h
5            Remove all training cases in T covered by ri.
6     else
7            Discard ri and remove it from R
8    end
9  Next r
```

Figure 3.3: FC pruning method

3.2.2 Two Criteria Pruning

In this section we present two pruning methods that combine two criteria when evaluating the candidate rules:

3.2.2.1 Full & Partial Coverage (FPC)

This is a combination of the FC and the PC pruning procedures in which each candidate rule is checked by whether there are training cases that fully match the rule. In cases where the candidate rule is not fully contained within any training case then the PC pruning will be invoked in which any partial matching is accepted. Lastly, in cases where there is no full or partial similarity between the candidate rule body and any training case the rule will be discarded. It should be noted that in the hybrid pruning of FPC the class similarity condition is relaxed.

When applying the FPC method on Table 3.2, rule ID (r_{12}) and considers it against the training data, four cases (3,7,12,13) are covered by this rule, those cases will be deleted and from the training data set and the rule is inputted into the classifier. The next ranked rules IDs (r_{13}, r_{15}) are applied and both cover certain numbers of training cases these are (1,2,8,9,11) so they will be added into the classifier and their cases will be removed from the training dataset. The procedure proceeds with rule ID (r_{14}) and found out that this rule is pruned since it does not cover any cases.

The next rules IDs (r_{11}, r_8) cover cases (5, 6, 10), (4) respectively and therefore both get added to the classifier and their cases get removed. The procedure then picks rules ($r_{9,}$ $r_{10,}$ r_7) and found out that these rules have no training cases coverage so it discards them. Lastly, one case left uncovered in the training case (case 14) and rule ID (r_5) and at this time the training data becomes empty. Thus, the FPC output the classifier which represents six rules.

The FPC pruning procedure terminates when all rules are tested or the training dataset is empty.

```
Input: Training data set $T$ and Set of Ranked Rules $R$
Output: classifier $h$

1  $\forall$rule $r \in R'$
2    if $r$ fully match  a training case then
3          Insert the rule at the end of $h$
4                 Remove all cases in $T$ covered by $r_i$
5          else if $r$ partially match at least a single case then
6                 insert the rule at the end of $h$
7                 Remove all cases in $T$ covered by $r_i$
8                   else
9                 Discard $ri$ and remove it from $R$
10 Next $r$
```

Figure 3.4: FPC Pruning Method

3.2.2.2 Full & Partial Coverage Correctly (FPCC)

The FPCC pruning procedure shown in Figure 3.5 is similar to FPC hybrid method with a slight difference that FPCC pruning invokes the full match or the partial match that necessitates class similarity between the rule and the training case. When a rule such as R is evaluated, FPCC checks whether there is training cases that can be covered with R (R's body is fully within any training case), if the test turns up to be true then R will be inputted into the classifier and all of its related cases get deleted. If the test turns up to be false, the partial matching procedure of Section 3.2.1.2 gets invoked and applied against the training cases, and we repeat the same process for the next ranked rule.

After applying the FPCC pruning on Table 3.1 it turns out that the resulting classifier contains six rules similar to the FPC pruning procedure. These results are restricted to this example only (Table 3.1) and often both methods produce different results. Finally, this pruning procedure terminates when all rules are evaluated or all training cases are covered.

```
Input: Training data set $T$ and Set of Ranked Rules $R$
Output: classifier $h$
1  $\forall$rule $r \in R'$
2    if $r$ fully match  a training case and Correctly classify it then
3          Insert the rule at the end of $h$
4          Remove all cases in $T$ covered by $r_i$
5      else if $r$ partially match at least a single case then and the class is matched
6          insert the rule at the end of $h$
7          Remove all cases in $T$ covered by $r_i$
8            else
9                  Discard $ri$ and remove it from $R$
10   Next $r$
```

Figure 3.5: FPCC Pruning Method

Table 3.3 lists the rule that have been inputted to the classifier after applying each one of the proposed pruning procedures which already has been discussed in each section above. Also the final classifier is described. The impact of applying each pruning method will be demonstrated in section 3.5.1

Table 3.3: selected rules when applying the proposed pruning on the data in Table 3.2

RuleID	**PC**	PC[3]	**FC**	**FPC**	**FPCC**
12	3, 7, 12, 13	3, 7, 12, 13	3,7,12,13	3,7,12,13	3,7,12,13
15	1, 2, 4, 8, 9, 11, 14	9,11,4	1,2,8	9, 11	9, 11
13	5,6,10	1,2,8 6	9,11	NA	NA
14	NA	NA	NA	5,6, 10	5,6, 10
11	NA	5,10	5,6,10	4	4
8	NA	NA	4	NA	NA
9	NA	NA	NA	NA	NA
10	NA	NA	14	NA	NA
7	NA	NA	NA	14	14
5	NA	NA	NA	NA	NA
6	NA	NA	NA	NA	NA
4	NA	NA	NA	NA	NA
1	NA	NA	NA	NA	NA
2	NA	NA	NA	NA	NA
3	NA	NA	NA	NA	NA
Total Rules in *Cl*	3 rules	4 rules	6 rules	5 rules	5 rules

3.3 Impact of rule pruning on classification accuracy

As mentioned before, AC employs association rule mining techniques in discovering the frequent itemset in transactional databases, where often the latter generates large numbers of potential rules. Dealing with such dense data without sitting some constraints on the rule discovery and generation steps or invoking appropriate pruning skills often results in a very large numbers of rules. Discarding redundant, noisy and those rules lead to wrong classification and preventing them to take any role in classifiers becomes essential.

Some associative classification algorithms that employ database coverage such as CBA (Liu, et al., 1998) prefer general rules over the specific ones and often they produce small classifiers (in term of the number of rules participating) unlike other techniques that employ Lazy Pruning such as L^3 (Baralisi, et al., 2002) which often form big classifier.

Consider for instance the experimental results against two UCI datasets to test the impact of rule pruning on classification accuracy by a lazy pruning algorithm of L^3, the (database coverage, pessimistic error) approach of CBA and our proposed methods. Results are generated using *minsupp* of 2% and *minconf* 40%. The numbers of rules produced by Lazy pruning of L^3 on the "Cleve" and "Diabetes" datasets are 6999 and 9847, respectively, with classification accuracies of 85.15% and 78.39%, respectively. Database coverage of CBA derives only 74 and 40 rules from the same datasets with Classification accuracies of 82.8% and 74.5%, respectively while one of our pruning that adopt partial coverage derives 108 and 96 from the same datasets with classification accuracy of 83,72% and 81,32%. The abovementioned results indicate that approaches employing database coverage and/or pessimistic error pruning tend to select general rules and form small classifiers which are need low computational cost but less accuracy (Liu, et al., 1998) (Thabta, et al., 2004).

Generally; small classifiers are preferred by human experts due to the fact that they are easy to understand and maintain. However, small classifiers suffer from some drawbacks, including, their sensitivity to datasets that contain redundant information and missing values and their lack of ability to cover the whole training data. Lazy pruning methods used in L^3 which produce large classifiers can slightly enhance the classification accuracy in some cases but on the expense of the computational cost and time in both, learning and prediction phases due t the fact that Lazy pruning method keep the spare rules to cover any test cases not covered by the primary set of rules. There should be a trade-off between the classifiers' size and the classification accuracy. Our proposed pruning methods take both aspects into consideration, they tend to choose specific rules and produce smaller classifier when comparing with those of lazy pruning and often slightly larger than those produced by approaches uses database coverage and/ or pessimistic error.

3.4 The proposed prediction approach (DC)

Predicting the class labels of a previously unseen data (test data) is the primary aim for classification task in data mining. Generally, predicting the class labels of test dataset in AC can be categorised into two main groups, The prediction procedure based on one rule such as those used in CBA (Liu et al., 1998) and MCAR(Thabtah, et al., 2005) works as following, the first rule applicable to the test case classify it. Whereas it is based on group of rules in other algorithms including CMAR (Li, et al., 2001) and CPAR (Yin X, et al., 2003) where scoring based methods are employs for group of rules before predicting the test cases. The main advantage of using group of rules for prediction is that there is more than one rule contributing to the final decision, which greatly limits the chance of favouring a single rule to predict all test cases satisfying its condition. However, algorithms that are rely on multiple rules in classifying test cases such as CMAR and CPAR do not consider the independence of the rules (Clark and Boswell, 1991) (Hu and Li, 2005), since during the training phase; cases are allowed to be covered by several rules with different class labels; this may cause rules dependency and conflicts. In other words, when a training case *t* is used to generate many rules during the rule discovery, then it is possible that in the prediction step, more than one rule with different class labels could be applicable to a test case similar to *t*. algorithms that utilise one rule in the prediction step may sometimes produce good classifiers, but assume that the highest precedence rule is able to predict the majority of test cases satisfying its body. In this section, we discuss the proposed prediction method.

The proposed prediction method will be discussed in this section along with an example to consolidate the idea. Consider the set of rules in table 3.4 that have been derived by MCAR (Thabtah et.al, 2005) with *Minsup* and *Minconf* of 20% and 40% respectively. The basic idea of the proposed prediction method that shown in Figure 3.6 is to choose the majority class among the set of high confidence, representative and general rules in the set of rules *R*. In classifying a test case (line 1), the proposed method counts the class label of all rules that fully match the test case body (line 7), and then using the class with the largest count (line 10) to Ts. In cases where no rule matches the Ts condition, the default class will be assigned to the test Ts (line 8).

```
Input: Classifier (R), test data set (Ts), array Tr
Output: Prediction error rate Pe

Given a test data (Ts), the classification process works as follow:
1 ∀ test case ts Do
2    Assign=false
3 ∀rule r in the set of ranked rules R Do
4 Find all applicable rules that match ts body and store them in Tr
5    If Tr is not empty Do
6         If there exists a rule r that fully matches ts condition
7             countperclass +=1
8 else assign the default class to ts and assign=true
9      end
10     if assign is false assign the dominant class to ts
11      empty Tr
12        end
13       compute the total number of errors of Ts;
```

Figure 3.6: Dominant Class prediction method

Table 3.4: A rule-based model

ID	Rule	confidence
1	Sunny& Hot-Temp→ Stay in	100%
2	Hot-Temp → Stay in	92%
3	Sunny → Stay in	76%
4	High-Humidity→ Go out	65%
5	Its-windy & Rainy→ Stay in	65%
6	Not-Windy→Go out	50%
7	Its-Windy→ Think	50%

Table 3.5: Test case

ts	Hot-Temp, Its-Windy, Sunny, Low-Humidity

To describe this method, consider the test case shown in Table 3.5, the subset of rules that are applicable to *ts* is shown in Table 3.6.To classify *ts*, we count the applicable rules per class, we found that "stay in" class has the largest count so we predict class "Stay in" for ts.

Table 3.6: Applicable Rules for *ts*

ID	Rule	confidence
1	Sunny & Hot-Temp→ Stay in	100%
2	Hot-Temp → Stay in	92%
3	Sunny →Stay in	76%
4	High-Humidity→ Go out	65%
7	Its-Windy→ Think	50%

Following section discusses the empirical study and analysis for the derived results.

3.5 Evaluation and Experimental Results

In this section, different pruning methods are compared with the proposed pruning methods (discussed in section 3.2)with respect to the classifier size and accuracy rate. A number of different datasets from the UCI data repository have been used in the experiments. Three pruning methods are compared with the proposed pruning methods: Database coverage (Liu, et al., 1998) lazy pruning (Baralis and Torino, 2002) and pessimistic error estimation (Quinlan, 1987). The bases of comparison are the number of rules and accuracy rate derived by the AC algorithms these are L^3 (lazy pruning method), MCAR (database coverage pruning method) and CBA (pessimistic error and database coverage pruning method).

3.5.1 Results and Discussion

Table 3.7 depicts the number of rules derived when the proposed pruning methods and three different pruning approaches are employed. The results of the proposed pruning methods in Table 3.7 have been derived using new enhanced version of MCAR algorithm called MCAR2 (Chapter 4). Table 3.7 indicate that algorithms which employ lazy pruning like L^3, generate larger number of rules due to keeping rules that do even cover a single training case in the classifier as well as adding rules which have been never selected during the learning phase to the spare rules into the classifier. The database coverage eliminates the spare rules and that explains its moderate size classifiers. Specifically, MCAR and CBA algorithms generate reasonable size classifiers if compared with L^3, which enables users to benefit from.

Moreover, CBA algorithm, which utilises database coverage and pessimistic error

Table 3.7: Number of rules produced by the proposed pruning methods and other pruning methods

Data	(Database Coverage, Pessimistic Error)	Database Coverage	Lazy pruning	The proposed methods				
				PC	PCCC	FC	FPC	FPCCC
Breast	47	67	22183	60	69	63	77	78
Cleve	74	102	6999	56	108	107	103	105
Diabetes	40	93	9847	35	96	99	102	106
Glass	29	39	11061	21	43	41	47	47
Heart	43	80	40069	29	81	83	85	85
Iris	5	15	190	33	15	17	16	17
Labor	17	16	7967	12	21	23	27	27
Led7	44	158	5860	91	154	176	162	166
Pima	40	93	9842	38	89	97	103	103
Tic-tac	28	28	41823	13	28	31	35	35
Wine	11	51	40775	9	56	61	65	65
Zoo	5	9	380921	5	9	12	15	15

pruning methods, cuts down further the size of the classifiers. Alternatively, the proposed pruning methods take both aspects, classifier size and classification accuracy into consideration, which tend to choose specific rules and produce smaller classifier when comparing with those of lazy pruning and often slightly larger than those produced by approaches uses database coverage and/ or pessimistic error.

Table 3.8 shows the classification accuracy for database coverage, and pessimistic error (CBA), database coverage alone (MCAR), Lazy pruning (L^3) and our proposed pruning methods of MCAR2 against number of datasets from UCI repository. The results revealed that the competitive classification accuracy was obtained using the proposed rule pruning methods.

One of the key reasons behind this appears to be that the proposed algorithm often derived more rules than the other ones. This also support (Baralis and Torino, 2002) in their conclusions concerning those algorithms that use database coverage and / or pessimistic error estimation are often discards some useful knowledge . The increase in classification rate suggests that removing unnecessary rules not only reduce rules redundancy but also justify the slightly growth of prediction power for the proposed algorithm that employed the new pruning methods over other AC algorithms.

Table 3.8: Classification accuracy after applying the proposed pruning methods and other pruning methods

Data	(Database Coverage, Pessimistic Error)	Database Coverage	Lazy pruning	PC	PC^3	FC	FPC	FPCC
Breast	95.85	96.1	95.99	96.3	**96.77**	96.3	95.8	95.79
Cleve	82.8	81.62	**85.15**	83.91	83.72	82.7	81.5	81.54
Diabetes	74.5	78.96	78.39	80.06	**81.32**	78.4	78.5	78.53
Glass	76.53	77.6	77.57	78.37	77.79	79.3	77.6	77.2
Heart	82.95	84.74	82.96	84.56	**85.62**	85.2	83.4	83.33
Iris	93.91	95.49	93.33	94.61	95.99	**96.7**	93.9	93.91
Labor	**94.99**	89.92	92.98	90.01	89.93	88.7	84.3	84.32
Led7	69.47	71.96	72	70.33	73.87	72.4	71.1	71.05
Pima	75.23	77.8	77.99	77.65	**78.93**	77.5	77.3	76.32
Tic-tac	98.76	99.23	99.48	**100**	**100**	**100**	99.5	99.45
Wine	94.96	95	97.19	97.43	96.98	**97.7**	95.5	95.54
Zoo	**97.78**	95.15	93.07	96.43	95.77	96.5	94.5	94.53
Average	**86.48**	**86.96**	**87.18**	**87.47**	**88.06**	**87.62**	**86.08**	**85.96**

The won-tied-loss records of PC method against database coverage & pessimistic error of CBA, database coverage, and lazy pruning methods are (10, 0, 2), (8, 0, 4), (8, 0, 4) respectively with +0.99, +0.51, +0.29 increase in accuracy respectively.

On the other hand, the PC^3 won-tied-loss records against the above scholars are (10, 0, 2),(10, 1, 1),(9, 0, 3)respectively and with +1.58, +1.1, +0.88 accuracy increase. Further, the won-tied-loss records of FC and FPC methods against the above mentioned methods are (9, 0, 3),(9, 0, 3),(8, 1, 3) and (7, 1, 4),(2, 1, 9),(4, 2, 6) respectively with +1.14, +0.66, +0.44 and -0.4, -.88, -1.1 in accuracy rate. Finally, FPCC won-tied-loss records against the above mentioned methods are (7, 2, 3), (2, 1, 7), (4, 1, 7) respectively with

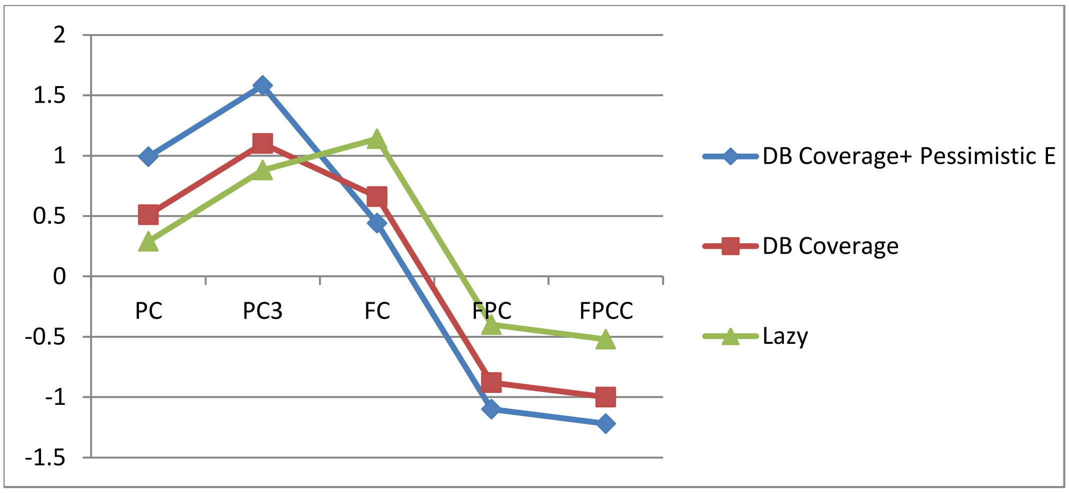


Figure 3.6: Accuracy rate for the three scholars against the proposed pruning methods

accuracy effect by -0.52, -1.0, -1.22 respectively. Figure 3.6 demonstrates the impact of the proposed pruning on the classification rate by showing the relative classification rate for the above scholars against the proposed pruning methods.

Referring to Figure 3.6 which depicts the accuracy results when employing the proposed pruning methods and other algorithms, Although FPC and FPCC pruning methods scores lower than those of CBA, MCAR but still they shows a competitive accuracy rates when contrasting with the above scholars. Further, the time taken to build the classifier when employing FPC and FPCC is less than time taken in MCAR. In other words, these two pruning methods scores a bit low in the accuracy rate but scores high in the running time and this is due to the fact that each rule covering more training cases and thus learning step finished faster; as we know, the learning process is stops either when no rules are left or all training cases are covered. Figure 3.7 depicts the average time taking (in ms) in building the classifier when employing the proposed pruning method and that of MCAR (Database coverage) when applying on a number of datasets form UCI.

Referring to the research questions mentioned earlier in this chapter and in the light of revealed results, it should be noticed that considering the rule if its antecedent is partially matched will have no negative accuracy impact on the final classifier since only the significant rules are selected and inputted in the classifier. Database coverage based methods such as (Liu, et al., 1998) (Thabtah, et al., 2005) (Qiang Niu et al. ,2009) prefer the general rules and require a full matching between the rule's body and the training case, class matching as well. Alternatively, PC and PC^3 pruning methods consider a rule

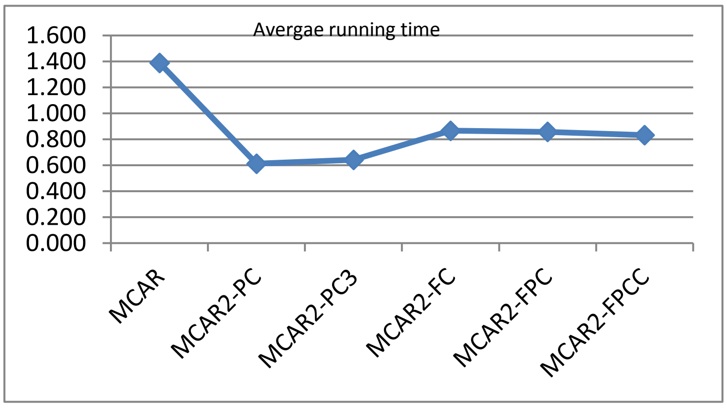


Figure 3.7: average running time of the proposed pruning methods and Database Coverage

significant if its condition partially cover the training case.

According to the results revealed in Table 3.8 we can see that the most of the proposed pruning methods have better impact on the classifier accuracy than database coverage and pessimistic error estimation which results in constructing competitive classifiers with respect to Classification rate as well the time taken in constructing them.

3.6 Summary

The numbers of rules that can be generated in the training phase of AC mining often large and accordingly lead to some drawbacks like hard maintenance of the classifier and user interpretation. There are two issues that must be addressed in this case. One is that such a large number of rules could contain noisy information which would mislead the classification process. Another is that large set of rules would make the classification time longer, this could be an important problem in applications where fast responses and accurate prediction are required. In this chapter we have introduced different rule pruning methods in AC to develop a competent AC model that achieves medium seizes classifiers with competitive classification rate. Moreover, a new class assignment procedure has been proposed which employs a group of rules prediction to further improve the decision of class prediction by a single rule.

We conducted experiments on a number of datasets selected from UCI repository with reference to the numbers of rules generated and accuracy rate using the MCAR2 algorithm (Chapter 4). As per the revealed results in the previous sections, PC^3 pruning method has outperformed all other methods including the database coverage, the pessimistic error, and the lazy and other proposed methods.

Next chapter introduces the proposed AC model where PC^3 pruning method and the dominant class prediction method are employed.

CHAPTER FOUR

MCAR2: AN ENHANCED MULTI-CLASS CLASSIFICATION BASED ON ASSOCIATION RULES

4.1 Introduction

Although recent research works showed some superiority of AC over traditional classification approaches many AC algorithms including CBA (Liu, et al., 1998), CMAR (Li et al., 2001), MCAR (Thabtah et al., 2005) and (Baralis, et al., 2008) tend to produce large numbers of rules. This is mainly because the learning mechanism employed by AC that use all relationships among attribute values and the class value to discover regularities in a form of simple If-Then rules. This may produce massive numbers of rules many of which are redundant, misleading or contradictory. The primary motivation of this work is to develop a new AC algorithm that contain appropriate pruning procedures in order to 1) cut down the numbers of rules derived in classifier and 2) not negatively impacting the classification rate of the classifier. The outcome is the new MCAR2 that contains a novel pruning and prediction procedures enhancing a known AC algorithm called MCAR. In Section 4.3 we highlight the main differences between the proposed algorithm and the original MCAR.

The proposed algorithm uses a new pruning method (Abumansour et al, 2010) that evaluates each rule by considering its coverage against the training data set and keeping only those that has training cases coverage without requiring class matching between the training case and the rule. We show later in 4.2.3 that this pruning method reduces over-fitting and generates less number of rules if contrasted with other AC algorithms like MCAR. Furthermore, MCAR2 not only reduces the classifier size by discarding unnecessary rules but also improve slightly upon the prediction power. In particular, the new prediction procedure overcomes problems associated with single rule prediction methods that utilise one rule to classify test cases satisfying its condition (Left hand side) during prediction step. In this context, the decision of class assignment of test cases will be based on multiple rules satisfying its condition instead of a single rule.

This chapter discusses the details of MCAR2 algorithm including rule discovery, rule sorting, rule pruning, and prediction procedure. The proposed algorithm intends to show the impact of rules pruning on classification accuracy and intends to enhance the classification rate and efficiency of AC algorithms by dealing directly with these two problems.

4.2 The MCAR2 algorithm

The proposed algorithm (shown in Figure 4.1) consist of two main phases, rules production and classifier construction. In the first phase, MCAR2 scans the training dataset a single time to find frequent 1 ruleitem and recursively combine ruleitems found in the current step to derive frequent 2 ruleitems, and so on. Any frequent ruleitem that has enough confidence is created as a rule. Once all rules are generated, a sorting procedure is used to discriminate among rules according to different rule criteria like rule's confidence, support, and length. Finally, the classifier is constructed from the set of sorted rules in which only rules which has an appropriate training data cases are stored in the classifier.

MCAR2 algorithm deals with nominal and continuous attributes in which continuous attributes are treated using a discretisation technique. In this context the Entropy based discretisation method (Fayyad and Irani, 1993) has been employed. Briefly, all training instances associated with a continuous attribute are sorted in ascending order along with their class values that linked with each instance. Then, break-points are placed at every time the class value is changed and to calculate the Information gain (IG) (Witten and Frank, 2000) for each possible break-point. The IG signifies the amount of information needed to assign values of the classes given breaking-points. At last, the break-point that minimises the IG over all possible breaking-points is chosen and the procedure is started again for the entire attribute values.

The frequent ruleitems discovery and rule generation of MCAR2 works as follows: Considering the training samples in Table 4.1. MCAR2 discovers the frequent ruleitems from the input data set and then from these frequent ruleitems it generated the class association rules (CARs) which are simply If-Then rules. For discovering frequent itemset, our algorithm scans the training dataset to find frequent 1 ruleitems, those that consist of only a single attribute value. Frequent 1 ruleitems are used for the discovery of potential frequent 2-ruleitems, and frequent 2 ruleitems are input for the discovery of potential frequent *3*-ruleitems and so forth. The frequent ruleitems procedure terminates once no more frequent ruleitems are discovered from the training dataset. According to Table 4.1, and with *MinSupp* equals 20% and *MinConf* equals 70%, the frequent 1- ruleitems set are: < <(*Att1, C*), *CL1*>, (*Att1, C*), *CL2*> < (*Att1*, *D*), *CL1*>, < (*Att2*, *T*), *CL1*>, < (*Att2*, *T*), *CL2*>, < (*Att2*, *Y*), *CL1*> with support frequencies (3,2,3,3,2,3) respectively. There is no frequent 2-ruleitems in this example since after

```
Input: Training data set (T), MinSupp and MinConf
Output:  classifier (Cl)

Scan T for the set $F_1$ frequent 1- ruleitem
$R \leftarrow F_1$
$j \leftarrow 1$
While $(F_j \neq \Theta)$
{
    $F_{j+1} \leftarrow Generate(F_j)$ (Figure 4.2 for the generate function)
    $TempCl \leftarrow TempCl \cup F_{j+1}$
    $j \leftarrow j+1$
}
$\forall$ Item(s) $\in F$
Generate rules as Antecedent→C
Rank (TempCLS) (Figure 3).
Cl $\leftarrow$ Evaluate TempCl on T (Figure 4.6)
Output CLS
```

Figure 4.1 MCAR2 algorithm

Table 4.1: example of a training data

RowID	Att1	Att2	Class
1	*C*	*T*	Cl_1
2	*C*	*X*	Cl_2
3	*C*	*T*	Cl_2
4	*C*	*X*	Cl_1
5	*D*	*T*	Cl_2
6	*D*	*T*	Cl_1
7	*D*	*Y*	Cl_1
8	*C*	*Y*	Cl_1
9	*D*	*Z*	Cl_1
10	*E*	*T*	Cl_1

Joining frequent 1 ruleitem and counting their frequency in the training data set (Table 4.1) all have failed to survive the *MinSupp* threshold and got discarded. The generation of the rules is a straightforward process that relies on the *MinConf* threshold once the complete frequent ruleitems have been discovered. The rules generated from Table 4.1 are $<(Att_1, D) \rightarrow Cl1$ and $<(Att_2, Y) \rightarrow Cl_2$.Those simply represents correlations among attribute values and class values that hold confidence above the *MinConf* threshold. This is the second goal of the algorithm.

Once the set of rules are generated from the frequent ruleitems as shown in the example above, MCAR2 invokes the sorting procedure shown in Figure 4.3 to discriminate among rules. Then, it selects a significant subset of rules to construct the classifier which latterly will be tested in predicting the class labels of test cases.

4.2.1 Training Dataset Representation

Data in AC can be represented in one of two layouts including vertical and horizontal. Many previous AC algorithms (Liu et al. ,2001) (Yongwook and Lee, 2008) (Kundu et al., 2008) (Niu et al. ,2009) have used the CBA (Liu, et al., 1998) horizontal data layout. There are fewer AC algorithms which have adopted the vertical data layout, e.g. (Thabtah et al., 2005) (Li et al., 2008). A data in horizontal layout contains a group of records, where each record has a unique ID and a list of objects contained in that record (Liu, et al., 2001). Having this format in an algorithm requires multi passing (scans) over the training data set when finding the frequent itemset at each level, this may lead to a highly computation cost (Zaki and Gouda, 2003) (Thabtah, et al., 2010).

Unlike data in horizontal format, databases in vertical layout consist of a collection of columns that contain an item followed by a list of row identifiers stored in a data structure known as tid-list that simply contains the item's occurrences in the training data set. Empirical studies including (Zaki and Gouda, 2003) (Thabtah, et al., 2005) (Li X. et al., 2008) showed that the vertical layout is more efficient in representing an input data because support counting of ruleitems are facilitated by fast intersections between the ruleitems' tid-lists. A study by (Li X. et al. , 2008) revealed that for long transaction databases, the vertical format reduces the number of I/O operations. Another study (Vo et al,. 2009), which investigates the integration of different AC approaches with database systems, revealed that vertical approaches are shown to perform better with reference to I/O time than horizontal ones. Despite the advantage of vertical data representation, when the cardinality of the tid-list becomes very large, intersection time become longer, this happens particularly for large and correlated transactional databases (Zaki and Gouda, 2003). Vertical data layout has been adopted to represent Training and classification data sets in MCAR2. A detailed example of data transformation using vertical data representation is shown in the next section.

4.2.2 Frequent Ruleitems Discovery and Rule Production

MCAR2 employs an intersection method called vertical mining adopted from MCAR (Thabtah, et al., 2005) for generating the complete set of ruleitems. The algorithm iterates over the training data set to count the frequencies of 1-ruleitems, from which it finds those that passes the *MinSupp* constraint. During the scan, frequent 1- ruleitem are determined, and their tid-lists are stored inside a data structure in a vertical format. Any ruleitem that fails to pass the *MinSupp* threshold is discarded. MCAR2 utilises the (Generate Function) shown in Figure 4.2 to find frequent ruleitems of size *k* by merging disjoint frequent itemsets of size *k*-1 and intersecting their tid-lists. The result of an intersection between the tid-lists of two ruleitems gives a new tid-list, which contains the row numbers where both *ruleitems* appear together in the training dataset. This new tid-list can be used to compute the support and confidence of the new ruleitem resulted from the intersection.

```
Generate Function
Input: A set of ruleitems S
Output: A set $S'$ of produced ruleitems

  $S' \leftarrow 0$
  Do
    For each pair of disjoint items $I_1, I_2$ in S Do
      If (<$I_1 \cup I_2$>, c) passes the minsupp threshold
          if (<$I_1 \cup I_2$>, c) passes the minconf threshold
              $S' \leftarrow S' \cup (< I_1 \cup I_2 >, c)$
          end if
      end if
    end
  end
  Return $S'$
```

Figure 4.2 MCAR2 Rule discovery algorithm adopted from [18]

Consider for instance the following items form Table 4.1, *<(Att1,C)>*, *<(Att2,T)>*; their occurancies are represented by the followng sets {1,2,3,4,8} and {1,3,5,6,10} respectivly. The new items' i.e. 2 rule item < *(Att1,C,), (Att2, T)>* suport can be determined by getting the intersection of the tid-lists sets for the items *<(Att1,C)>*, *<(Att2,T)>*, the resulting set *(1,3)* represent the rows where both items occure together in the training dataset. If the support of the 2 rule item passes the *MinSupp* threshold then its a candiate for a rule's condition. Those items will pass the *MinSupp* will recursivly be generated for those itesm that have a maller number of attributes starting from 1-rule item generated in the first pass.

To find the support for a ruleitem, we use the tid-list of its items to locate classes associated with it in the data structure (an array) and select the class with the largest frequency. So if an item is associated with two classes we choose the class that has larger representation with the item in the training data even though if the second class has survived the *MinSupp* requirement. Then the ruleitem support is obtained by considering the length of the tid-list set where the itemset and its largest class count and dividing it by the training dataset size, we can obtain the ruleitem support.

The confidence is calculated similarly except that the divisor is the size of the tid-list of the ruleitem’s antecedent (its items) only. Frequent ruleitems are produced recursively from ruleitems' conditions with a smaller number of attributes i.e. *K-1* starting with frequent 1 ruleitems which derived in one scan throughout the whole training dataset.

Considering the example of data given in Figure 4.3 which represents the training data set given in Table 4.1 in vertical data representation with *MinSupp* and *MinConf* of 20% and 50%, respectively. In the first pass, the frequent 1itemsets that pass the *MinSupp* threshold are identified, these are *(<ATT1, C>, <ATT1, D>,<ATT2, T>,<ATT2, X>,<ATT2, Y>)* and all other infrequent itemsets are discarded. 2 itemsets candidates are then produced by merging disjoint frequent 1 itemsets as shown in Figure 4.4. Once these itemsets are identified, we check their supports and confidences simultaneously and finally allocate classes where they occurred.

For example, for the 2 itemset candidate < (ATT1, C) (ATT2, X)> we locate its classes inside the class array using its rowIds {2, 4}. We select the class with the highest count, which either *Cl_1* or *Cl_2*. Assume we chose *Cl_1*, and divide the

Att1		
C	D	E
1	5	10
2	6	
3	7	
4	9	
8		

Att2			
T	X	Y	Z
1	2	7	9
3	4	8	
5			
6			
10			

Figure. 4.3 Vertical data transformation for the training dataset in Table 4.1

cardinality of the set {4} by the training dataset size, to calculate the support value of the *ruleitem <(Att1, C) (Att2, X), c_1>*. If it has sufficient support, we then calculate its confidence as explained earlier. For ruleitem < *(Att1, C) (Att2, X), c_1>*, the support and confidence values are 1/10 and 1/2 respectively. Now when a ruleitem survives the *minconf* threshold, we directly consider it as a potential rule to be in the classifier. In this particular example, ruleitem < *(Att1, C) (Att2, X), c_1>* is pruned before calculating its confidence since it did not pass the *MinSupp* threshold.

CT	CX	CY
1	**2**	8
3	**4**	

DT	DX	DY
5		7
6		

Figure. 4.4 Possible frequent 2-itemsets generated from Table 4.1

Most of current AC algorithms including (Liu, et al., 1998) (Yin and Han, 2003) (Baralis, et al., 2004) (Harnsamut and Natwichai, 2008) produce frequent ruleitems and obtain their confidences in two different steps whereas the proposed algorithm obtain them both in one step.

4.2.3 Rule Sorting and Building the Classifier

MCAR2 algorithm sorts rules based on the rule ordering procedure of MMAC (Thabtah et al, 2004) by adding one additional criterion called "class distribution" frequency in the training data set to distinguish among rules as shown in Figure 4.5. An experimental study performed in (Thabtah et al, 2005) revealed that in cases where there are very large number of rules that may have identical confidence, support and antecedent length will make the decision to favour among the rules very hard, this has to invoke the default class during the prediction step in many positions which may slightly degraded accuracy rate of the classifier. The rule ranking method employed by the proposed algorithm intends to discriminate among rules by using multiple criteria as indicated above. For two rules, r_a and r_b, with the same support, confidence and rule length, but r_b is linked with a class that has larger frequency in the training dataset than that of r_a, the algorithm favours r_b over r_a during the ranking. In some cases rules r_a and r_b might have the same support, confident, rule length and class frequency, in such cases the choice is arbitrary.

Given two rules, r_a and r_b, r_a precedes r_b ($r_a \rangle r_b$) if:

1. The confidence of r_a is greater than that of $r_{b.}$
2. The confidence values of r_a and r_b are the same, but the support of r_a is greater than that of $r_{b.}$
3. Confidence and support values of r_a and r_b are the same, but r_a has fewer conditions in its left hand side than of $r_{b.}$
4. Confidence, support and cardinality of r_a and r_b are the same, but r_a is associated with a more representative class than that of $r_{b.}$
5. All above criteria are identical for r_a and r_b, but r_a was generated from items that have higher order in the training data than that of r_b.

Figure. 4.3 MCAR2 Rule sorting procedure

After having the set of all rules extracted from the training dataset and ranked in descending order according to the above procedure, MCAR2 forms the classifier as follows:

For each sorted rule, the algorithm starts with the first rule and checks its applicability to the training cases (data coverage), the rule is added in the classifier if it covers at least one item in training case regardless of the rule's class similarity to that of the training instance. The class similarity between the candidate rule and the training instance does not necessarily indicate the rule significance besides the coverage condition between this rule antecedent and the training instance.

Once a rule gets marked as a classifier rule according to the above procedure, all of the training cases associated with it are removed from the training dataset. In situations where a rule fails to cover any training case then it will be discarded. The process is iterated until no more cases remains in the training dataset or all candidate rules are tested. Finally, all marked rules get inserted into the classifier. The evaluation procedure in our algorithm (shows in Figure 4.6) aims to keep only high confidence and quality rules in the final classifier.

4.2.4 Class assignment Procedure

Unlike some AC algorithms such as MCAR and CBA which utilise one rule for predicting the class label for test instances, the proposed algorithm predicts the class label based on multiple rules prediction. Previous studies in AC have considered the multiple rule in the prediction step such as (Li et al,. 2001) .

The significance of making prediction decision based on multiple rules lies in that more than one rule participating in the prediction decision will significantly narrow the chance of using one rule to predict all test cases that satisfying its antecedent.

```
Input: Training data set T and Set of Ranked Rules R
Output:  Classifier h
1  R' = sort(R);
2  ∀rule ri in R'
3     Find all applicable training cases in T that match ri's condition Where
      at least one item of ri's condition in ti
4       Insert the rule at the end of h
5          Remove all training cases in T covered by ri.
6  else
7                            Discard ri and remove it from R
8    end
9  Next r
```

Figure 4.4 the pruning Procedure in MCAR

However, algorithms that are rely on multiple rules in classifying test cases such as CMAR (Li et al ,. 2001) and CPAR (Yin and Han, 2003) did not consider the rules independency. When a training instance *t* is used in generating several rules during the rule discovery phase, it is likely that multiple rules with different class values could be applicable to a test case similar to *t*.

Figure 4.5 explains the novel class prediction method called "Dominant Class" used in MCAR2. In classifying a test case, the new prediction procedure divides all rules which fully match the test case antecedent (contained in the test case attribute values) into groups according to their class labels. In other words, all rules applicable to the test case *t* are grouped by class values, and then assign the test case the class of the dominant group (the group which has largest count of rules). In cases when no rules in the classifier are applicable to the test case, the default class (Majority class in the training dataset) will be assigned to that case.

```
Input: Classifier (R), test data set (Ts), array Tr
Output: Prediction error rate Pe

Given a test data (Ts), the classification process works as follow:
1 ∀ test case ts Do
2    Assign=false
3 ∀rule r in the set of ranked rules R Do
4 Find all applicable rules that match ts body and store them in Tr
5    If Tr is not empty Do
6         If there exists a rule r that fully matches ts condition
7             countperclass +=1
8 else assign the default class to ts and assign=true
9      end
10     if assign is false assign the dominant class to ts
11      empty Tr
12        end
13       compute the total number of errors of Ts;
```

Figure 4.5: Dominant Class prediction method

The choice of full match between the rule body and the test case attribute values is due to the fact that the algorithm is looking for the best matching rules which signify the chance of correct classification. Moreover, the decision of class assignment of the test case has been voted by more than one rule in most cases which increases the confidence in the prediction decision unlike single rule prediction algorithms.

In the pruning step, the discovered rules are evaluated on the training data set in order to remove insignificant ones and full matching principle here produces very accurate classifier if used but only on the training data set which may over-fit the training data set and therefore the performance of the classifier might be poor elsewhere (unseen data), this justifies the use of partial matching in the pruning step but not in the class assignment. Meaning, the aim of classification in data mining has not been achieved by just testing the classifier on training data set since the classifier already knows it very well. It is like teaching someone how to drive a vehicle in a small village like "Queensbury-West Yorks" in UK for three months and then when it comes to testing his driving skills the tester takes him to Queensbury in which the driver knows every road, pumps, curves, etc. Most likely he will pass the test easily, though if the tester wants to generalise him as a good driver then he may let him perform the driving test in cities like Leeds which the driver never drove in. That's why we have used partial matching while evaluating the rules in pruning and full matching when it comes to prediction.

4.3 Features of the proposed Algorithm

MCAR2 algorithm has some distinctive features over the existing AC algorithms as follows:

- Most AC algorithms adopt horizontal data representation where multiple scans over the database are necessary to discover the frequent items and generate the rules. The proposed algorithm adopts vertical data representation and a recursive learning procedure by interesting the frequent items to discover rules which require only one pass over the database to do the task. SPRINT (Shafer, et al., 1996), which is a traditional decision tree technique, uses a similar data format to vertical layout to store attribute values called attribute lists. However, it does not use fast intersections of rowIds to discover the rules, instead it builds decision tree similar to traditional decision tree algorithms (Mehta, 1996). MCAR algorithm employs vertical data layout however it stores both the items tid-lists and the class tid-lists separately unlike MCAR2 which stores them together.

- Some AC techniques, e.g. CBA, MCAR consider a rule significant during building the classifier if it's fully and correctly cover a training instance. MCAR2 employs a new rule evaluation which considers the rule significant if it's partially covers training cases. Experimental test against different classification benchmark problems conducted in Section 4.4 show that the proposed algorithm produces competent classifier with good classification rate and less size.

- Most of CBA based AC algorithms utilise a single rule class assignment for test cases. There could be more than one rule applicable to a test case with similar confidence (Li et al ,. 2001); (Yin et al ,. 2003) .The rule with highest confidence order may not be effective especially when it applied on datasets with unbalanced distribution of class labels (Liu et al., 2003). The proposed algorithm uses a new prediction method that classify the test cases using multiple rule (details are given is section 4.2.4)

4.4 Experiments

In this section, different rule-based classification algorithms are compared with MCAR2 according to the classification rate, and the classifier size (Number of rules). 18 different datasets from the UCI data repository (Merz and Murphy, 1996) have been used in the experiments. The algorithms used in the comparison are: C4.5 (Quinlan, 1993), RIPPER (Cohen W. 1995), MCAR (Thabtah, et al., 2005), and the proposed algorithm. The reason behind selecting these algorithms is the different training strategy they employ in discovering the rules. C4.5 uses information gain (IG) in in the induction of tree. Each path in the decision tree from the root node to the leaf denotes a rule. The IG measures how well the each attribute is suited with the class. If the attribute value is associated with a single class the gain value for this attribute will be 1.

RIPPER is a search algorithm that utilises an exhaustive searching strategy to build the rules. It starts with the training dataset and divides it into two sets, one is the pruning set and other is the growing set rules. The rule growing set starts with an empty set and then the algorithm heuristically adds one condition (an attribute value) at a time to the rule until the rule has zero error rates on the growing set. The algorithm repeats the same steps until the growing set becomes empty. RIPPER uses the pruning set to removes duplicate rules during building the classifier.

MCAR and MCAR2 are an AC mining algorithms that scan the training data set to count the frequencies of 1-ruleitems, from which it finds those that passes the *MinSupp* constraint. Both algorithms apply tid-list intersection to discover the remaining ruleitems. Once this step is completed, MCAR and MCAR2 employ rule sorting and pruning to construct the classifier.

Ten-fold cross validation (Witten And Frank 2000) is applied as a testing mechanism to derive the classification rate.

The experiments of the proposed algorithm and MCAR have been run on Pentium IV machine with 2.0 Gb RAM and 2.3 Gh processor. MCAR and our algorithm have been developed in Java and tested locally. RIPPER and C4.5 algorithm results were generated from (Weka, 2001).. This software tool contains implementations of collection of data mining algorithms such as classification, filtering, association rule mining, regression and rule induction.

Table 4.2 represents the datasets details which include the number of cases, the number of attributes, the number of classes, and the class distribution for each dataset.

One of crucial parameters in AC is the *MinSupp* since it controls the number of rules generated. Empirical studies (Thabtah et al ., 2005) (Liao et al ., 2009), concluded that setting the *MinSupp* high may result losing important rules, and setting it low may produce numerous rules. There is no research works that pointed out the optimum value of the *MinSupp* threshold since each data set has its own characteristics. Therefore, following (Liu et al., 1998) (Li et al., 2003) (Thabtah 2010) in setting the *MinSupp* threshold to values between 2% and 5%, we choose 5%.

Table 4.2 Datasets Details

Dataset	No. Of attributes	No. Of classes	Class Distribution	No. of cases
diabetes	8	2	65% 35%	768
glass	10	2	76% , 24%	214
heart	13	2	56%, 44%	270
iris	4	3	33%, 33%, 33%	150
labor	16	2	65% 35%	57
pima	8	2	65% 35%	768
Led7	8	10	10%, 10%, 10%, 8%, 11%, 10%, 11%,9%, 10%, 10%	3200
tic-tac	9	2	65% 35%	958
wine	13	3	33%, 40%, 27%	178
zoo	18	7	41%, 20%, 5%, 13%, 4%, 8%, 10%	101
Austral	14	2	45% , 55%	690
breast-w-699	10	2	65% , 35%	699
Cleve	14	2	58%, 42%	303
Mushroom	22	2	52%, 48%	8124

The *MinConf* on the other hand has no high impact on the rules derivation process and therefore it has been set to be 40% similar to (Thabtah et al., 2005) (Harnsamut, et al., 2008) (Yang et al., 2009).

4.4.1 Results and Analysis

Table 4.3 shows the classification rate for RIPPER, C4.5, MCAR and MCAR2 against the 18 UCI datasets. The results clearly show that MCAR2 algorithm outperform the remaining algorithms in term of accuracy. On average, MCAR2 algorithm achieved +0.62%, +0.39% and +1.72% higher prediction rate than RIPPER, C4.5 and MCAR respectively on the datasets we consider.

Table 4.4 lists the won-tied-lose records of the proposed algorithm against C4.5, RIPPER and MCAR when it comes to average classification rate on the datasets. The experimental results indicate superiority of MCAR2 when contrasted with other algorithms.

Table 4.3: Accuracy of C4.5RIPPER, MCAR and MCAR2

Dataset	**C4.5**	**RIPPER**	**MCAR**	**MCAR2**
Australian	76.23	78.800	83.304	**87.145**
Breast	85.21	**95.100**	92.475	91.130
Cleve	**94.56**	77.550	75.875	75.870
Contact	73.33	75.100	75.000	**77.583**
Diabetes	73.82	**74.700**	71.992	73.798
German	70.91	69.000	69.130	**74.910**
Glass	66.82	68.660	64.486	**73.178**
Heart-s	76.95	78.230	76.633	**78.946**
Iris	92.15	94.000	92.200	**95.133**
Labor	73.68	77.200	69.825	**78.772**
Led7	**73.56**	69.700	70.472	72.897
Lymph	**81.08**	77.100	77.446	69.865
Mushroom	**99.77**	99.100	95.565	98.194
Pima	72.78	73.100	71.979	**73.763**
Tic-tac	83.71	96.000	**99.948**	98.653
Vote	**88.27**	86.140	81.517	86.667
Wined	**94.38**	91.600	80.730	83.450
Zoo	93.06	85.140	**97.604**	87.545
Average	81.68%	81.45%	80.35%	**82.07%**

Table 4.4 shows the won-tied-loose records of accuracy when comparing the proposed algorithm with the remaining algorithms on the datasets we considered.

Table 4.4: won-tied-loss records of the proposed algorithm

	C4.5	RIPPER,	MCAR
MCAR2	10-1-7	12-0-6	13-1-4

The slightly higher prediction rate of MCAR2 over the remaining algorithms is due to the multiple rules prediction procedure used by the algorithm. Unlike MCAR, C4.5 and RIPPER which employ a single rule prediction procedure that takes the first rule class that satisfies the test case to classify that case, MCAR2 chooses the class that belongs to the class that has the largest count of rules. In other words, the proposed algorithm is benefited from using multiple rules decision and significantly limits the chance of preferring a single rule.

Another possible reason for the slight improving in the classification rate of MCAR2 is the rule sorting procedure that limits the use of the default class during prediction by imposing new criteria which is the class frequencies of rules when discriminating among rules. We have run an experiment to show the difference in accuracy between using three criteria and four criteria in rule sorting within MCAR2, the results have showed that when utilising the class frequency as the forth tie breaking condition in

rule sorting on average, for a sample of 10 data sets MCAR2 increases 0.47%. Figure 4.8 shows the enhancement on accuracy rate after using the rule ranking procedure i.e. 4 conditions in the proposed algorithm over the 3 condition procedure against 10

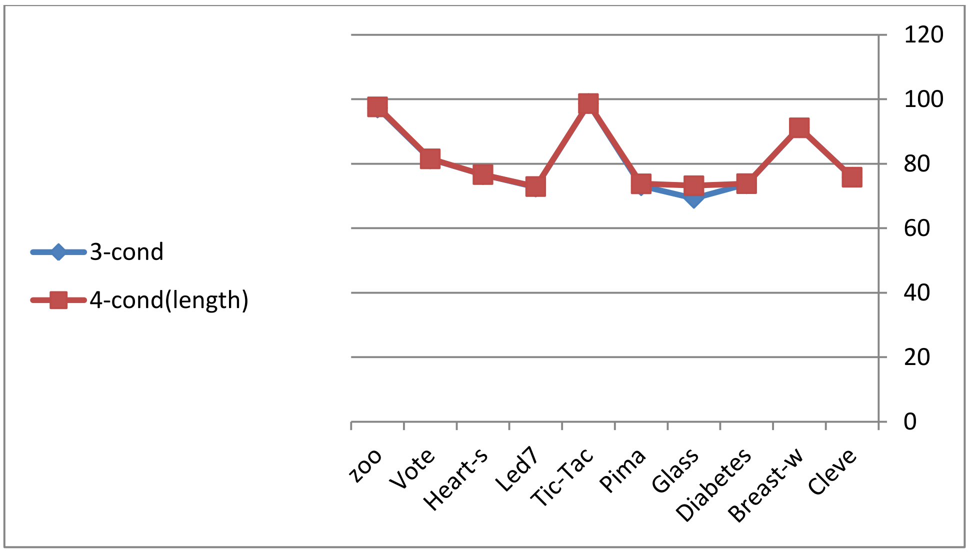


Figure 4.6: The enhanced accuracy rate after using the 4 ranking conditions against 10 datasets

datasets from UCI. The won- tied- loss records for the 4 conditions ranking over the 3 conditions ranking is 7-3-0. This gives an indicator that using random selection when two rules have the same support confident and length is not a proper decision for all cases.

Figures 4.9a, 4.9b and 4.9c shows the "relative accuracy rate" that denotes the variation in the accuracy rates of the proposed algorithm with reference to those resulting by C4.5, RIPPER and MCAR. It indicates how much good or bad MCAR2 performs with reference to C4.5, RIPPER and MCAR learning techniques on the datasets used. The relative accuracy rate details given in Figures 4.7a; 4.7b and 4,7c are conducted using the following formulas:

$$RR = \frac{(Accuracy_{\ MCAR2}) - (Accuracy_{\ target})}{(Accuracy_{target})} \qquad \text{equation (4.1)}$$

For example, the relative accuracy rate of MCAR2 algorithm on “Lymph” dataset is negative since MCAR2 achieved a lower classification rate than C4.5, MCAR while the RR is positive for the majority of the datasets since the proposed algorithm achieved higher relative prediction accuracy than the rest of the algorithms.

Figure 4.8 shows the number of rules derived by MCAR and MCAR2 which clearly indicates that classifiers with moderated size may positively impact the classification accuracy.

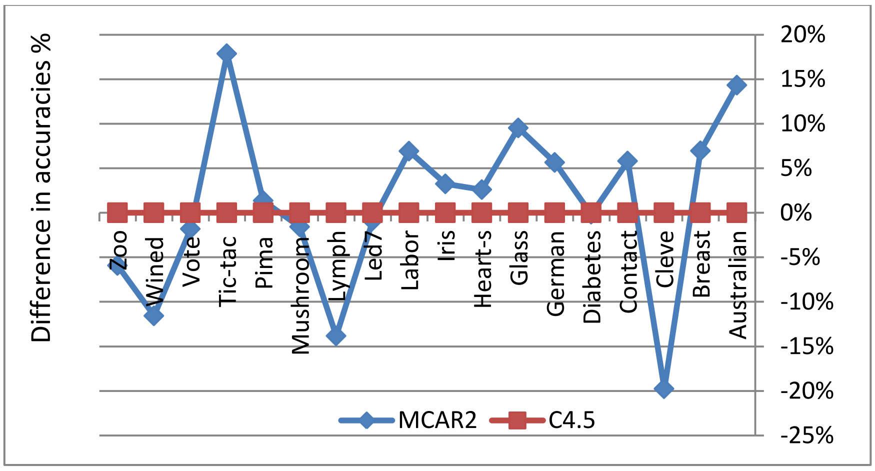


Figure 4.7a Difference of accuracy between C4.5 and MCAR2

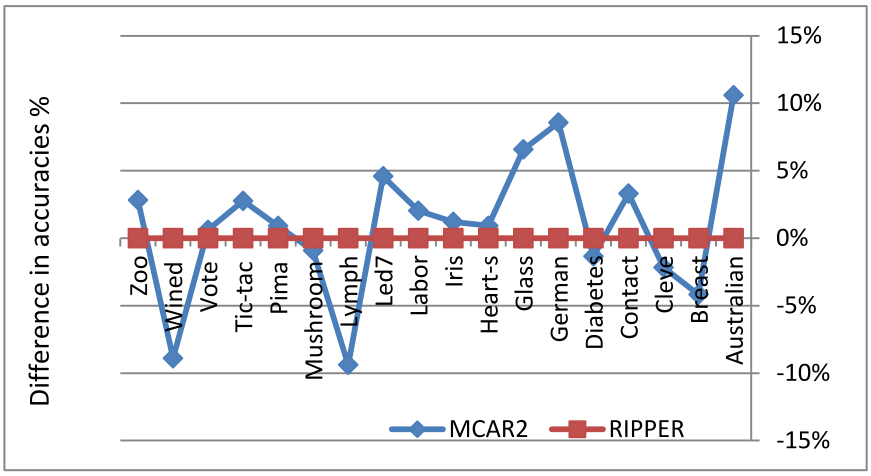


Figure 4.7b Difference of accuracy between RIPPER and MCAR2

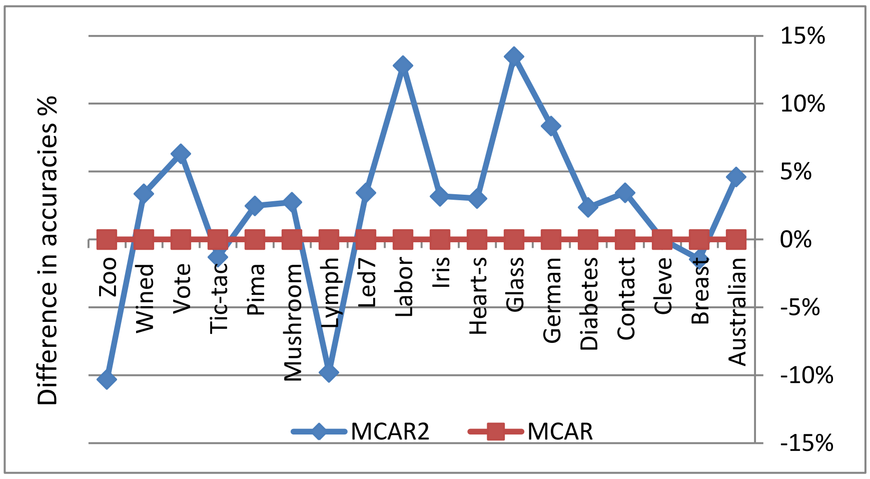


Figure 4.7c Difference of accuracy between MCAR and MCAR2

For the 18 datasets, Figure 4.9 and on average the proposed algorithm derives 13.87% less rules than those derived by MCAR algorithm. Thus, MCAR2 algorithm compromises between producing the classifier size and the classification rate in a way that it generates highly competitive classifiers yet smaller in size if compared with the remaining

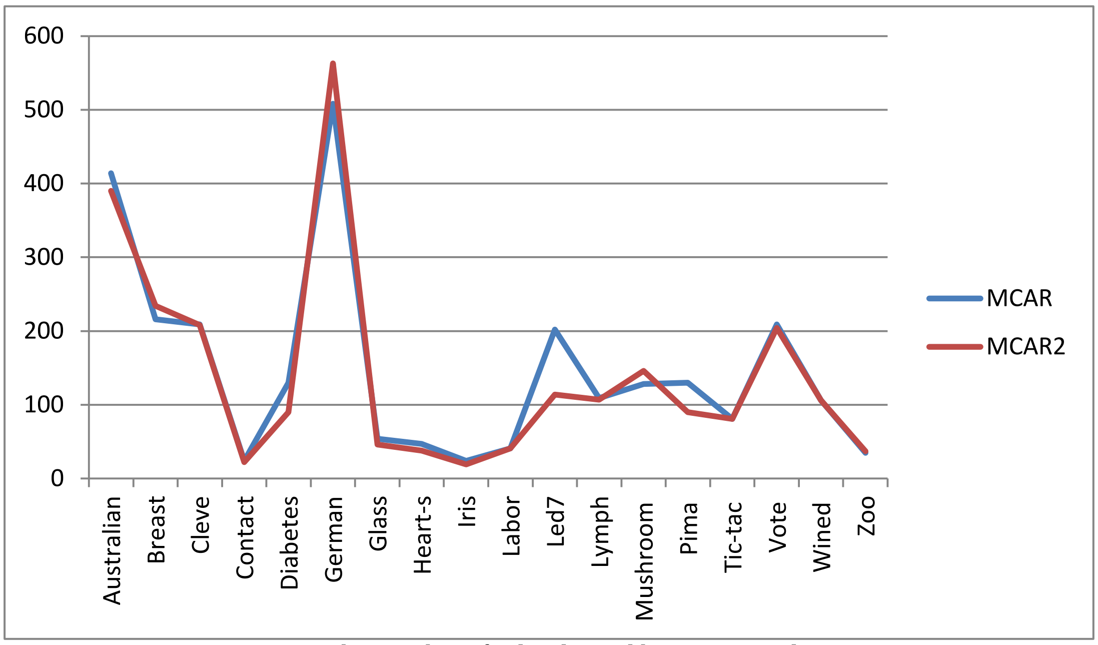


Figure 4.8: The Number of rules derived by MCAR and MCAR2

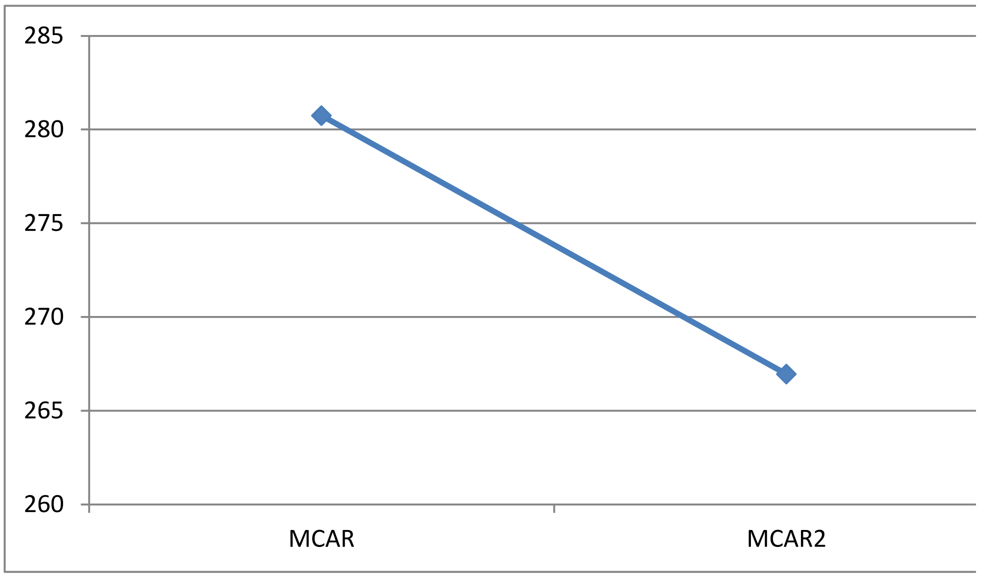
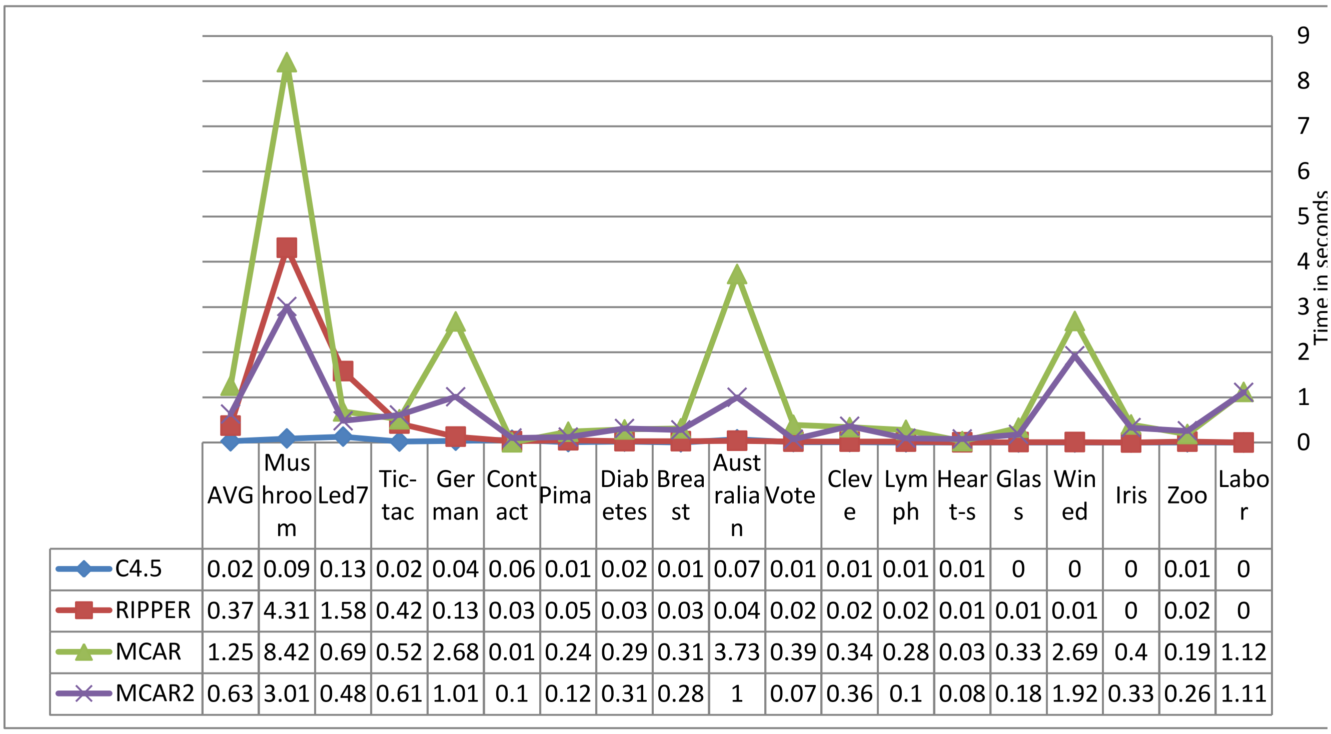


Figure 4.9: The average number of rules generated by MCAR and MCAR2 on 18 datasets

The processing time taken to build the model in the proposed algorithm has been compared with those of RIPPER, C4.5 and MCAR in order to evaluate the efficiency and scalability on the UCI 18 dataset. In this part we are going to investigate principally whether MCAR2 reduces the learning time taken to build the model when contrasted with that of MCAR. Figure 4.10 shows the Processing in seconds extracted in the experiments. The processing time taken to build the model when using different algorithms is displayed in Figure 4.10. The processing time results reveal that MCAR2 is faster than original MCAR in most cases; the won-tied-loss records of MCAR2 against MCAR are 13-1-4. This is due to the fact that MCAR2 is employing partial matching when evaluating the potential rules during building the classifier step and this greatly covers more training dataset. AC algorithms that employ the intersection technique avoid multiple database passes, therefore they require less time than those who employed multiple passes such as Apriori and FP-growth.



	AVG	Mushroom	Led7	Tic-tac	German	Contact	Pima	Diabetes	Breast	Australian	Vote	Cleve	Lymph	Heart-s	Glass	Wined	Iris	Zoo	Labor
◆ C4.5	0.02	0.09	0.13	0.02	0.04	0.06	0.01	0.02	0.01	0.07	0.01	0.01	0.01	0.01	0	0	0	0.01	0
■ RIPPER	0.37	4.31	1.58	0.42	0.13	0.03	0.05	0.03	0.03	0.04	0.02	0.02	0.02	0.01	0.01	0.01	0	0.02	0
▲ MCAR	1.25	8.42	0.69	0.52	2.68	0.01	0.24	0.29	0.31	3.73	0.39	0.34	0.28	0.03	0.33	2.69	0.4	0.19	1.12
× MCAR2	0.63	3.01	0.48	0.61	1.01	0.1	0.12	0.31	0.28	1	0.07	0.36	0.1	0.08	0.18	1.92	0.33	0.26	1.11

Figure 4.10: Time taken in building the Model

The processing time results indicate that classical classification algorithms like C4.5 and RIPPER are faster than AC methods on the majority of the datasets we considered. This is due to the simple structure and due to the fact that C4.5 and RIPPER is using many pruning skills during the classifier construction process. On the other hand, AC algorithms are using classic association rule mining techniques in the rule learning step, which requires more computational time in discovering the frequent itemset as well as generating the rules

4.5 Summery

In this chapter, the problem of association rule mining techniques has been investigated. A new efficient classification algorithm MCAR2 has a number of new features over other existing AC algorithm has been presented. The proposed algorithm uses a detailed rule ranking method, which adds a new significant breaking condition that considers the distribution frequency of class labels in the training dataset to favour one rule over another. This new condition has proved its effective in reducing the use of random selection since it has been used frequently in many experiments against classification benchmark (see section 4.4). In rule discovery, MCAR employs a an intersection strategy for ruleitems tid-lists that requires only one database scan, consuming less processing time than those learning methods which require more than one pass over the database. More importantly, the proposed algorithm has a novel pruning procedure i.e. Partial coverage (section 3.2.1.1) that reduces the number of rules in AC mining significantly and a new multiple rules class prediction procedure that overcomes any slight decrease of the accuracy during pruning.

Performance studies on 18 data sets from UCI data repository showed that the proposed algorithm is highly competitive when contrasted with classical classification algorithms such as RIPPER and C4.5. The proposed algorithm shows better performance if contrasted with existing popular AC approaches like MCAR with respect to classification rate, rules significance, classifier size and effectiveness. Experimentations using 18 correlated classification problems indicated that using additional constraints to break ties between rules improve the accuracy rate of the resulted classifiers.
The proposed algorithm employs new pruning skills which consider a rule as significant rule if it does partially cover a training case regardless to the class value matching. To the best of our knowledge there is no AC algorithms used this partially covering in rule pruning. The proposed algorithm uses multiple rules prediction instead of one single rule. This has slightly increased the proposed algorithm average classification rate.

Next chapter expand the investigation on the impact of rule pruning on the accuracy rate by applying the proposed algorithm on text categorisation problem. Precisely, we checked the applicability of the MCAR2 on large complex and high dimensional unstructured data which usually produce a huge number of rules.

CHAPTER FIVE

The Application of the Proposed Model to Text Categorisation: A Case Study

5.1. Introduction

Text categorisation is the process of automatic assigning category labels for an un-labeled text documents. Automated text documents classification is an important application domain which attracted many researchers since the dense amount of digital documents all over the databases available online and offline. Text classifiers have to assist the information retrieval tasks and deal with large text data like those available in the web, scientific journals as well as other domains such as emails classification and memos. The main task of text classification system is to assign category label for new un-labeled documents. A number of different approaches for the text categorisation task are proposed in the literature. This includes Decision trees (Quinlan, 1993), Neural network classifiers (Wiener et al., 1995)., k-NN classifiers (Mitchell et al., 1996), Support Vector machine classifier (Joachims T, 1998) and (Joachims T, 2001), Naive Bayes classifier (Lewis et al, 1998), Regression techniques (Yang and liu. 1999), associative classifiers (chen et al., 2005) (Baralis et al ,m 2006) (Li et al,. 2007) and others.

Several research works including (Liu et al., 1998) (Li, et. Al, 2001) (Yin X, et.al, 2003) (Thabtah et al,.2005) provide evidences that AC approach generates highly competitive and scalable classifiers if contrasted with other classic classification approaches such as Rule induction and decision trees. However, those approaches were tested against small numerical and structured data from UCI repository (Merz, C., and Murphy. 1996) but not widely for text data and other unstructured data. This chapter aims to investigate the impact of employing associative classifier to build a model for text Categorization problem on the classification accuracy.

(Sebastiani, 1999) defines the problem in *Text Categorisation* as follow: The text data sets are divided into two typeset, training and testing documents, giving a training dataset *T=(d1,d2,d3,……dn)* where *n* is the number of documents to be used in constructing the classifier. These documents must have an accepted number of terms (words) that matches the given categories. *(d1, d2, d3,….dm)* is the testing dataset where they used to measure the classification process's accuracy. TC is a task to find approximation function to predict unknown target meaning: TC aims to form a classifier CL to predict unclassified document and can be formulated as the following function:
$CL: D \times C \rightarrow \{T, F\}$ where $C = (c_1, c_2 \ldots c_i)$ is the set of the predefined category labels and D= $(d_1,d_2,\ldots\ldots, d_j)$ is the set of the finite documents. Now if $CL(d_i, c_j)= T$ then d_i is truly classified by c_j or d_i member in c_j (Positive example) while $CL(d_i, c_j)= F$ *means* d_i is wrongly classified by c_j or d_i member in c_j (Negative example). Figure 5.1 demonstrates the TC problem:

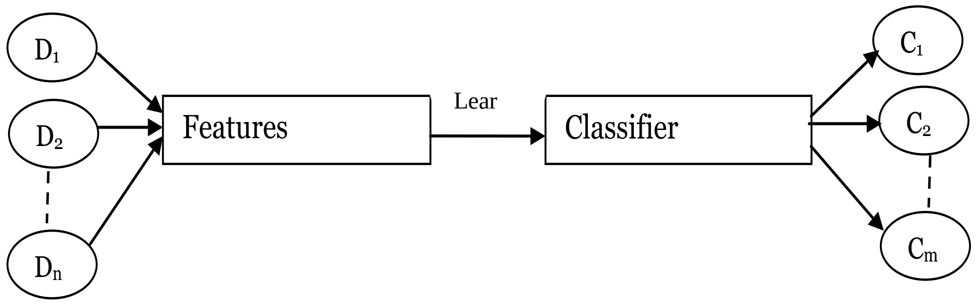


Figure 5.1: TC problem

5.2 Text Categorisation Phases

After collecting the dataset, three phases are involved in Text Categorization: the document indexing, classifier learning and construction and finally, measuring the classification process's accuracy and efficiency. Discussions on each phase are given in the following sections. Prior to these phases, a pre-processing procedure is invoked which Addresses converting the document to a representation to be suitable for the learning procedure. This procedure does a number of tasks such as removing HTML tags if we are dealing with web based documents, eliminating the stop words i.e. Prepositions and spatial characters …etc in preparations for tokenizing the document (segmenting the document in to words/ terms), and finally word stemming (converting each phrase into its root for example: categorisation will become category, computing will becomes compute),

5.2.1 Document Indexing and Dimensionality Reduction

Document indexing is the process of mapping a document *d* into representation of its content which can be interpreted either by the learning algorithm or the classifier after being constructed. Indexing methods are represented by identifying the term either as its occurrence in the text or with its stems (the term root)   or by computing the term weight which could be binary or non- binary valued using probabilistic or statistical techniques, the choice is upon the used algorithm.

TC often deals with dense unstructured data. Hence, most of TC algorithms necessitate the dimensionality reduction step in order to have smaller document representation which helps making the learning process more manageable. Dimensionality reduction usually takes the form of feature selection where a scoring function is applied for each term that denotes its correlation with the class label. Only terms with the highest scores are selected for document representation. The following sections demonstrate the common methods in indexing and dimensionality reduction.

5. 2.1.1 Indexing and Numerical Vector Representation

A number of proposed techniques for document indexing and representation and have been proposed in the literature, bag of Word Representation is the simplest way of representing the document into a suitable form for the learning procedure. *WR* by (Lewis David 1998) represents a document as follows: $D_{cj}=(t_{i1},t_{i2},t_{i3},\ldots t_{in})$ where D is the document, *T* is the set of terms and *i* is the index for each term. The methods will start with the pre-processing phase by segmenting (Tokenizing) the documents into words. This is done each time white spaces faced, and then the extracted words will go through stemming process i.e. returning each word into its root and then removing the stop words like prepositions and spatial characters. The output of this method is an unordered list of terms that represent a document. Term frequency (Tokunaga.et.al, 1994) is another technique used to evaluate the importance for a term *t* in document *d* by assigning a weight for each term found. The weight of term *t* is simply the number of its occurrences in a document and it is formulated by the following function:

W (t, d) = TF (d, t), weight *W* of a term *t* in a document *d* is a represented by the term frequency *TF* for *t* in *d*. Knowing the weight for each term can help in categorizing the text documents as considered as a frequent terms, Words that likely appear in many documents usually have less discriminative power for that term.

Unlike term frequency which searches the frequency for a term *t* in a single document *d*, Inverse Document Frequency (IDF) by (Sparck, 1972) measures the importance of a term among *N* documents contain that term. A term importance increases if it appears in few numbers of documents and decreases if it appears in large number of documents. In other words, for a given text dataset contain *N* documents; *n* will be the number of documents where term *t* is appeared. In data set contain 1000 documents, given two states for *t*, appear in 10 documents and appear in 80 documents, IDF is computed by the following the equation: IDF (t) = log (N/n)

The importance for t in the first state = log (1000/10) =2 and in the second state= log (1000/80) =1.1, which proofs for the theorem of the IDF.

IDF treated all documents where a term t is occurred equally since it is employing binary counting. IDF does not consider the number of occurrences of t in these documents, rather it considers the fact that t is occurred in these documents.

Weighted Inverse Document Frequency (WIDF) by (Tokunaga et al., 1994) is an extension of IDF (Inverse document frequency). *IDF* counts the occurrences of a term *t* regardless how many times it occurs in documents *d*. alternatively; *WIDF* has extended the *IDF* approach to incorporate the term frequency over the text collection. *WIDF* of a term *t* in document *d* is counted by:

$$WIDF(d,t) = \frac{TF(d,t)}{\sum_{i \in D} TF(i,t)}$$

Where *TF(d ,t)* is the occurrence of *t* in *d* and *i* is the number of the documents in dataset *D*, *WIDF* gives the frequency of a term t over the text corpus.

IDF approach has been criticized by (Lan, et al., 2006) by claiming that this approach has been proposed for the purpose of improving the discriminating power of the terms in traditional *IR* field but it may not be the case in the *TC*. For further explanation on this claim, let us discuss the example given in Table 5.1:

Table 5.1: Examples of three terms which share the same IDF value

Term	Sum (a,c)	a : c	idf value
t_1	100	10 : 1	log(N/100) = 3.322
t_2	100	1 : 1	log(N/100) = 3.322
t_3	100	1 : 10	log(N/100) = 3.322

Where *a* is the occurrence of a term *k* in the positive category and *c* is the occurrence of the term *k* in the negative category .For a given a category c_i, t_1, t_2 and t_3 are sharing the same *IDF* value. However, the details given in the table indicates that t_1 and t_3 have more discriminating power than t_2. (Lan, et al., 2006) attempted to improve the discriminating power by proposing a new term indexing method called relevance frequency, which is defined as the following equation:

$$rf = log(2 + \frac{a}{c})$$

Where *2* is a constant value, *rf* factor gives more importance to t_1 than t_2 and t_3 since t_1 contributes more to the positive category. The reason behind giving more importance to term which is assigned more in the positive documents than negative documents is due to the fact that positive documents belong to one category while the negative one are scattered on multiple categories.

5.2.1.2 Dimensionality Redaction

Feature selection is employed in selecting the best subset of association rules which focuses on the relevant data and reducing the high dimensionality of the features. Usually in text mining the feature selection techniques are either comprehensive or heuristics. In the comprehensive approach, all of the possible features are discovered and the best feature among them based on a certain criterion is considered. Such approaches are computationally expensive but often achieve better accuracy. In heuristics approaches, the selection is based on the score of each feature. Feature with highest score above predefined threshold is the higher relevancy to the document.

There are many approaches for feature selection such as Associative feature selection (Do et al, 2006), Information Gain (Lewis and Ringuette 1994) and chi-square X^2 (Snedecor et al., 1989) which they are examples on supervised approaches while Document Frequency (Yang and Pedersen 1997) and Term Strength (Wilbur et al., 1992), are examples on unsupervised approaches.

Chi-square can be used in dimensionality reductions and it's considered as a supervised method based on statistics. It evaluates the correlations between two features and decides whether they correlated or not (Snedecor et.al, 1989) and considers a certain number of the most the highly correlated feature according to the scores they gained. For each term *t* in category *c*, Chi-square, χ^2 is computed by using the following equation:

$$X^2(t,c) = \frac{N(AD-CB)^2}{(A+C)(B+D)(A+B)(C+D)}$$

Where *N*: is the total number of training documents, *A* is the number of documents in a category *c* containing *t*, *B* is the number of documents not in a category *c* containing *t*, *C* is the number of documents in a category *c* not containing *t*, *D* is the number of documents not in a category *c* not containing *t*.

Chi-square testing has been employed in many TC algorithms including (Caropreso et al.2001) (Galavotti et al. 2000) (Schutze et al.1995) (Sebastiani et al. 2000) (Yang and

Pedersen 1997) (Yang- and Liu 1999) and showed good performance as promising results.

Information Gain (IG) is another supervised approach used to measure and count the amount of the gained information for category prediction by testing the absence and occurrence of term *t* in document *d*. This is computed by using the following equation:

$$\mathrm{IG(t)} = -\sum_{i=1}^{m} P(c_i)\log P(c_i) + P(t)\sum_{i=1}^{m} P(t,c_i)\log P(t,c_i) + P(\bar{t})\sum_{i=1}^{m} P(\bar{t},c_i)\log P(\bar{t},c_i)$$

Where *m* is the categories count, *P(c_i)* is the probability of the category c_i, *P(t,c_i)* is the joint probability of the category c_i and the occurrence of the term *t, P(t)* is the probability that the term t occurred in a document, and $P(\bar{t}\ )$ is the probability that the term *t* is absent in a document *d*. IG shows good performance when applied to TC problem (Caropreso et al. 2001) (Larkey 1998) (Lewis and Ringuette 1994), (Mladeni´c 1998) (Yang and Pedersen 1997) (Yang and Liu 1999).

In (Do et al., 2006), an associative feature selection approach for text mining is proposed which unsupervised heuristic approach to split the set of terms into two sets (relevant and irrelevant terms). Meaning, two terms occurs in many association rules are given a high score and they are considered as a relative terms. Terms which are occurring in few association rules would have a low score. It is based on the relevancy of the associated features in the text documents. Features are extracted using the association rule mining. The set of generated rules will be evaluated by using relative confidence technique proposed in (Do et al., 2006). For rule $r = x \rightarrow c$, the *relative confidence* is computed by the following equation:

$Rconf(r) \quad = \frac{P[\ X\hat{}Y] - P[X]P[Y]}{P[X] - P[X]P[Y]}$ Where X and Y and two terms

The procedures of assigning the score to the features consist of three steps:

1) Defining the thresholds. Usually, support and confidence are the constraints used in data mining. In the proposed approach, the relative confidence threshold is used.

2) Generating the set of rules using one of the rules mining approaches and keeping only those satisfying the predefined thresholds. Here Apriori was used.
3) Scoring the features (association rules), based on their occurrences; terms which occurring in many rules are scored high.

5.3 Learning and constructing the classifier

Due to the rapid growth of the digital text documents to assign the text documents to one or more predefined categories, many classifications methods have been developed and applied to text categorisation for both, binary problem (the text document either classified as relevant or not relevant to a predefined categories) and multi class (where more than two categories in the corpus) and multi label problem (where more than two categories in the corpus and a text document could be relevant to one or more predefined categories). Several algorithms have been proposed in the literature

5.3.1 Naive Bayes

Naive Bayes is a famous probabilistic approach to classify test objects (Duda and Hart, 1973), NB has been applied on text categorization problem (Yang and Liu, 1999) (Yang, 1999) (Thabtah et al., 2009) (Hadi et al., 2008b).

Let *d* be a training document with no class label and *h* be the hypothesis/ assumption such that (*d* belongs to class *ci*). In classifying *d,* to describe *h* given the observed document *d, p(h|d)* (the probability of *h* given *d*). For example, the probability that a liquid is water, given the condition that it is black.

Naive Bayes calculate the probability *P(h|d),* from *P(h), P(d),* and *P(d|h)* by using the following relation : $P\ (h|d)\ = \frac{P(d|h)\ P(h)}{P(d)}$ where *P (h|d)* is the probability that *d* belongs to *h, P(h)* is the probability of a class *h* indicates the number of documents that belong to a category divided by overall number of documents and *P(d|h)* is the probability of document *d* given class *h.* One shortcoming of the Naïve Bayes algorithm is when attribute values do not occur for every possible class in the data set, the probability of such an attribute belonging to a class that has never occurred with it is zero. Since this fraction is multiplied by other probabilities, the final probability will be zero. A minor adjustment to the method of calculating the probabilities can be accomplished by adding a very small integer, say b to the fraction numerator and compensating with b/3 to the denominator (Witten and Frank, 2000). The Laplace estimator method (Snedecor and Cochran, 1989) offers another solution to such a problem, which adds 1 to the numerator and compensates by adding 3 to the denominator. Missing values are omitted in Naïve Bayes (Duda and Hart, 1973).

5.3.2 Decision Trees

C4.5 (Quinlan, 1993) is the most famous example of decision trees: C4.5 builds the decision tree from the training dataset. Let $T=(t_1,t_1,\ldots.t_n)$ be the set of training instances with known class. Each instance $t_i=(x_1,x_2,\ldots)$, $x_1,x_2\ldots$ is the set of attributes of t_i. The training instances are assigned to vector $C=c_1,c_2\ldots$ which represent the set of class labels.
For each node, one attribute is chosen which is the effectively splits its set of values into subsets augmented in one class label. The chosen is based on the information gain for each attribute. The attribute with highest *IG* value (Quinlan, 1986) will chosen to split the data, the process is repeated on the smaller sub lists.
C4.5 can be summarized into 4 steps:

1. For each attribute *X* find the IG from splitting on *X.*
2. Let *XH* be the highest attribute in term of IG criterion.
3. Build a decision *node* that splits on *XH.*
4. Repeat on the sublists obtained by splitting on *XH*, and add the resulted nodes as children of *node.*

Mitchell (1997) and Joachims (1998) applied C4.5 on TC. The results showed that the C4.5 produced competitive results if compared with other methods such as K-nearest Neighbor (Yang, 1999), Support Vector Machine (Vapnik, 1995) (Schapire et al., 1998).

5.3.3 Neural Network (NN)

Neural Network (Wiener et al., 1995) has a set of nodes divided into layers. The first layer is the input layer followed by zero or more middle layers, and an output layer. Each node receives an input weight and produces an output.

When applying Neural Network to text categorization, the first layer would contain the set nodes that contain the set terms and the output layer would contain the categories. In classifying a document *d*, the set of terms' weights will be stored in the input nodes. Then those input nodes will be broadcasted through the network middle layers until a result is found and sent to an input node. For further details, see (Li and Park, 2006).

5.3.4 K-nearest Neighbor.

K-NN (Yang, 1999) is a statistical approach used for classifying instances based on the k closed training cases. K-NN is applied in many fields of studies such as pattern recognition, data mining and text categorization.

In machine learning, K-NN is the simplest algorithm for classification. Basically, the test cases are classified by the majority weight of its neighbor. A test case is assigned to the most common class among its k nearest neighbors, where *k* is an integer number.

5.4 Evaluation Measures for the Text Classification Process

The performance for a TC algorithm can be measured either by its efficiency or effectiveness. Efficiency describes the time taken in the learning the classifier and/or the time taken to classify the test cases. Efficiency becomes very important when it comes to experimental comparison between different learning algorithms or different TC algorithm.. Algorithm's effectiveness describes the average classification rate. Effectiveness on the other hand tends to be the primary measure of performance of an algorithm

The best measurement criterion for the single label problem is the classification accuracy. However, the classification accuracy isn't favorable in binary and multi-label problems. This is because binary TC has two categories which are often unbalanced sine one contains much more than the other. This will lead to build classifier with high accuracy rate since most of the test cases will be assigned to the most heavily populated category. Hence, Binary and multi-label TC systems measured by a combination of precision and recall Sebastiani (2005)

Generally, for a given TC system, documents can be divided into four different sets Precision (*P*) and recall (*R*) are the effectiveness measurements in binary and multi-label TC systems. Use terminology from logic, *P* can be viewed as "degree of soundness" of the classifier *ci*. On the other hand *R* can be viewed as "degree of completeness". According to the above definition, *P* and *R* are subjective probability means the expectation of the human that the system will perform correctly in classifying test cases. Table 5.2 shows the estimation of these probabilities where TPi (True Positive) is the count of correctly

Table 5.2 Contingency table in TC for *ci*

Category		**True**	**False**
System Classification	**True**	*TPi*	*FPi*
	False	*FNi*	*FPi*

classified documents under c_i, FP_i(False Positive) is the count of incorrectly classified documents under *ci*, FN_i (False Negative) and FP_i (False Positive) are defined accordingly. Precision P and Recall R can be defined in the following relations:

$$P = \frac{|TPi|}{|TPi \bigcup FPi|} \qquad \text{And} \qquad R = \frac{|TPi|}{|TPi \cup FNi|}$$

Assume there are 5 blue and 7 red balls in a pool and you intend to retrieve the blue ones only. If you could retrieve 6 balls where 4 of them are blue and 2 are red. This means you have retrieved 4 out of 5 blue (1 false negative case) and 2 red (2 false positives cases). Accordingly, precision=4/6 (4 blue out of 6 retrieved balls), and recall= 4/5 (4 blue out of 5).

For multi-label problems, methods such as precision and recall need to be combined in order to measure the performance of all classes properly since the document might belong to more than one category. Therefore, a hybrid method, called *F1* by Rijsbergan (1979) that measures the average effect of both precision and recall together, has been applied in *IR* and *TC*. *F1* criterion for a given *P* and *R* is defined as the following relation:

$$F1(P,R) = \frac{2 * P * R}{(P+R)}$$

F1 is computed for each class independently and then the means of the results is computed on the test data set as a whole using one of two different methods named "*macroaveraging*" and "*microaveraging*" (Yang et al., 2002) in order to reflect the quality of the classifier. Macroaveraging represents the average of precision (recall) for all categories and microaveraging accumulates the decisions for all categories (summation for all true positives, false positives and false negatives cases), and then precision and recall are calculated using the global values. The *microaveraging of P* and *R* is given in relation 5.9 while *macroaveraging (μ)* of *P* and *R is* given in relation 5.10

$$P^{\mu} = \frac{\sum_{i=1}^{|c|} TPi}{\sum_{i=1}^{|c|} TPi \bigcup FPi} . \qquad R^{\mu} = \frac{\sum_{i=1}^{|c|} TPi}{\sum_{i=1}^{|c|} TPi \bigcup FNi}$$

$$P^{M} = \frac{\sum_{i=1}^{|c|} P}{|C|} . \qquad R^{\mu} = \frac{\sum_{i=1}^{|c|} Ri}{|C|}$$

The breakeven point (*BEP*) is another measurement (Joachims, 1998) it is the point where precision equals recall:

$$BEP(P,R) = \frac{P+R}{2}$$

Overall, TC researches, including (Joachims, 1998) (Yang and Liu, 1999) (Yang, 1999) (Antonie and Zaïane, 2004) (Yoon and Lee, 2008) (Thabtah et al., 2009) use error-rate (accuracy) method, Precision, Recall, and F1 to come up with the effectiveness of their classifiers.

5.5 Text Based Associative Classification

Associative classification integrates association rule mining and classification proved its efficiency on the numerical data. In the last decade, AC has been adapted to deal with many other applications such as detecting phishing websites, email phishing, Biometrics and text categorisation. A Pre-processing phase is invoked first in order to transform the text data which is often unstructured into a form to be suitable for the learning phase based on a derived numerical datasets to form a classifier. Figure 5.2 illustrates the Associative Text Classifier Model.

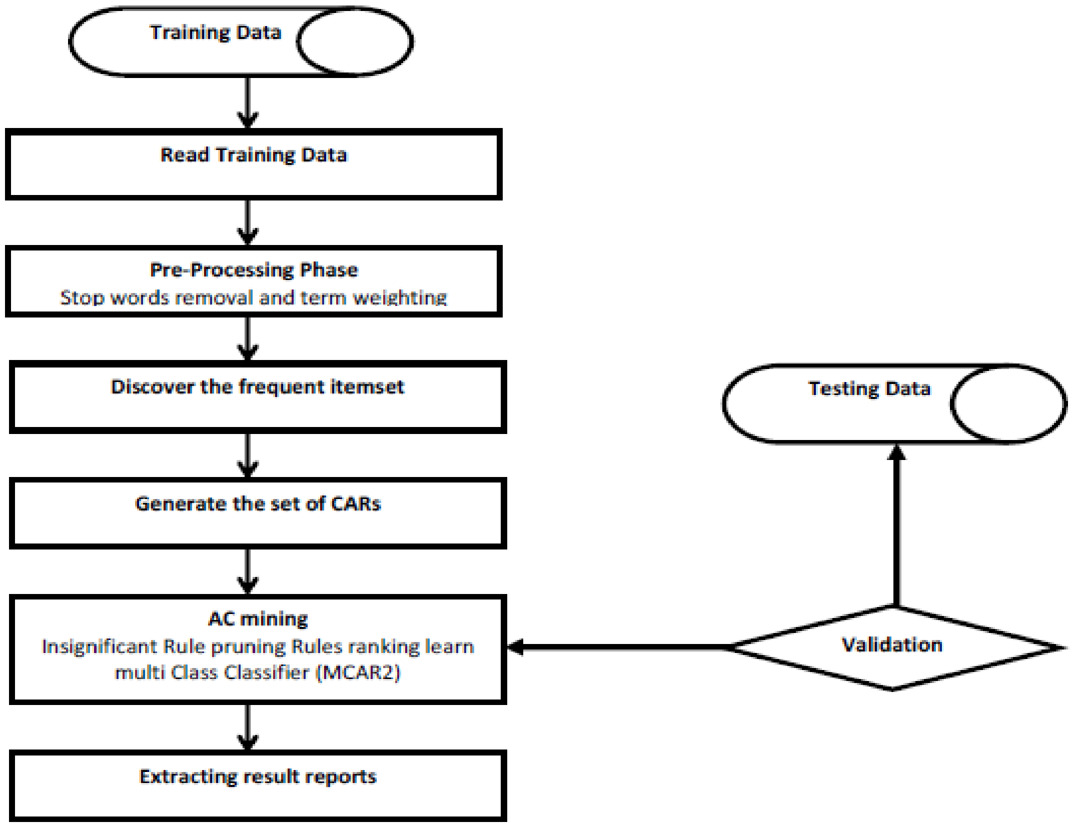

Training Data
Read Training Data
Pre-Processing Phase
Stop words removal and term weighting
Discover the frequent itemset
Generate the set of CARs
AC mining
Insignificant Rule pruning Rules ranking learn multi Class Classifier (MCAR2)
Extracting result reports
Testing Data
Validation

Figure 5.2: associative text classifier model.

In this section, the proposed AC model "MCAR2" will be adapted to TC problem. Section 5.6.1 shows the text corpus used in this study and demonstrates the experiment results on text data.

Association rules mining and associative classification approaches have been adapted to text categorisation problem in many research works; Antoni (2002) is one of the earliest studies that employed association rule mining approach to TC. There was an attempt to form a classifier to deal with TC problems by incorporating term co-frequency approach. Apriori has been adapted for discovering the frequent ruleitems and rule generation. Noisy and insignificant rules are discarded using database coverage method, whereas the prediction is done by assigning the class label with highest confidence. ARC-AC proposed by (Antonie et al,. 2002) is based on the Apriori algorithm and utilizes basic rule ranking. The algorithm evaluates the set of generated rules using database coverage method. ARC-AC is the successor of ARC-BC (Antonie et al., 2002), which adopts same mechanism except adopting a global approach that extracts rules by treating each category separately and combining them afterwards.

ACTC is another novel AC algorithm for TC problem based on correlation analysis proposed by (Chen et al., 2005). ACTC aims to extract the *K*-best correlated negative and positive rules directly from the training dataset in a way to pass up employed complex *Minsupp* and *Minconf* constraints. As an alternative of generating the set of candidate rules, the algorithm employed Foil-gain to evaluate the significance of generated rules and generate a small subset of the most predictive rules. Those rules with weak correlation score are discarded and only positive and negative rules which passed the evaluation procedure are left to be in the classifier. Experiments on Reuter's corpus against C4.5 show that ACTC perform better. ACTC keeps the good rules so call "close to the best rules".

BCAR (Yoon and Lee, 2008) is another algorithm that adapts AC to TC which generates a large number of association rules then rules derived are filtered using a method equivalent to a deterministic Boosting algorithm (Freund and Schapire, 1997). This pruning method is a modification of the database coverage pruning (Liu et al., 1998). The BCAR algorithm can be utilized in large-scale classification benchmarks like TC data. Experiments using various text collections showed that BCAR achieves good prediction if compared with SVM (Vapnik, 1995) and Harmony (Li et al., 2007).

.

MCFF is another TC algorithm proposed by (Srinivas et al., 2008). MCFF integrates the multi-type features co-selection procedure based on clustering and feature selection based on pseudo-class-based score selection. The objects are clustered in two groups and each cluster corresponds to a real category. Apriori is used to derive the set of rules which usually generate large numbers of rules. Database coverage evaluation procedure is applied to cut down the number of rules. Association rule –based classifier by category (ARC-BC) is used in classifier construction step and finally, Class assignment step is done as follows: all rules applicable to test case are grouped by category label and the group with highest confidence sum is assigned to that case.

5.6 Empirical Study and Experiments

Different traditional classification algorithms as well as rule-based classification algorithms are compared with MCAR2 according to the prediction rate. The benchmark used in the experiments is the Reuters-21578 (Lewis .D, 1998). The Reuters-21578 is the most commonly used text data set in the text categorisation research. We used the ModApte version of Reuters-21578 that leads to a corpus of 9,174 documents(6,603

Table 5.3 Number of documents in training and testing sets per category (REUTERS-21578)

Category	Training	Testing
Acq	1650	719
Crude	389	189
Earn	2877	1078
Grain	433	149
Interest	347	130
Money-FX	538	197
Trade	396	117

training and 2,571 testing documents). The algorithms used in the comparison are CBA (Liu, et al., 1998), BCAR(Yoon and Lee, 2008), MCAR (Thabtah et al., 2005), Naïve Bayes (Lewis et al, 1998) and K-NN(Mitchell et al., 1996). The experiments are conducted on PIV 2.3 Gh processor and Gb RAM. The proposed methods and MCAR are implemented using VB.Net programming language with a minsupp and minconf of 2%, and 40%, respectively. Table 5.4 shows the number of documents in training and testing sets per category (REUTERS-21578).

On these documents, the preprocessing phase was limited to stop word elimination and tokenizing but not stemming, and we selected the top 1000 features using Chi Square approach (Snedecor and Cochran, 1989) to reduce the feature space.

In the experiments we adopted the Macro breakeven point (BEP) evaluation measure (Joachims, 1998) as the base of our comparison; breakeven point (BEP) as the point where precision equals recall Equation.

In the Macro BEP, one contingency table per class is used; the BEP is computed for each table, and lastly all results are averaged. Table 5.4 depicts a comparison results between the classifiers produced by the proposed algorithm against other well-known Text Classifiers. It should be noted that the results of the BCAR algorithm are reported in (Yoon and Lee, 2008) and the results for the other classification systems are given in (Qian et al., 2005). For MCAR, we implement it and adapted to TC to derive its results.

Table 5.4: Precision/Recall-BEP for MCAR and other scholars on seven most populated Reuter's datasets

Category/Algorithm	Naïve Bayes	*k*NN	CBA	MCAR	BCAR	MCAR2
Acq	91.5	92	89.9	90.2	97.8	**99.5**
Crude	81	85.7	77	88.1	88.1	82.8
Earn	95.9	97.3	89.2	**99.8**	97.4	**98.8**
Grain	72.5	88.2	72.1	95.3	86.5	**98**
Interest	58	74	70.1	41.6	**83.5**	58.1
Money-FX	62.9	78.2	72.4	74.3	84.4	**92.7**
Trade	50	77.4	69.7	**96.2**	89.8	95.3
AVG	73.11	84.69	77.20	83.64	89.64	89.31

The results revealed that proposed algorithm outperformed the traditional and AC classification approaches we consider expect BCAR algorithm. Beside, Table 5.5 lists the

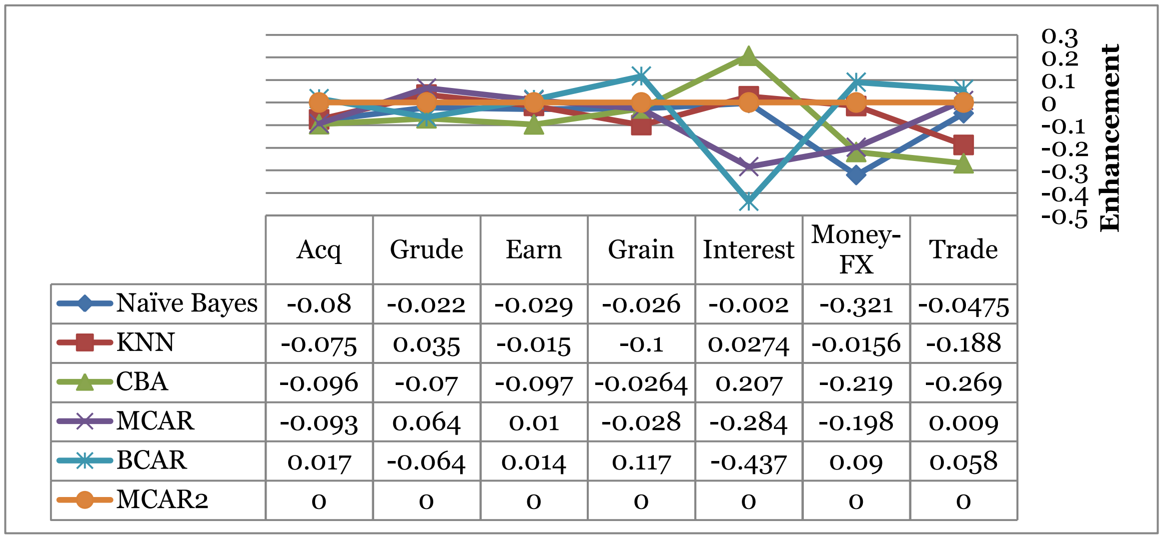
Table 5.5: won-tied-lose records of the proposed algorithm against the other algorithms

	Naïve Bayes	*K*NN	CBA	MCAR	BCAR
MCAR2	7-0-0	5-0-2	6-0-1	4-1-2	4-0-3

won-tied-lose records of the proposed algorithm according to the average classification rate on the datasets. The experimental results indicate superiority of MCAR2 and BCAR. We would like to justify that the slightly higher prediction rate of the MCAR2 algorithm over the remaining algorithms due to the multiple rules prediction procedure used by the algorithm and for BCAR was due to usefulness of the used rule selection approach which is close to AdaBoost algorithm (Freund and Schapire. 1997) that improves the training process and error generalization and the normalized score model used in predicting test cases.

Figures 5.3 shows the "relative BEF rate" that denotes the variation in the accuracy rates of the proposed algorithm with reference to those resulting by the above scholars. In other words, it indicates, how much good or bad MCAR2 performs with reference to above scholars learning techniques on the datasets used. The relative accuracy rate details given in Figures 5.3 are conducted using the following relations:

$$RR = \frac{(Accuracy_{\ MCAR2}) - (Accuracy_{Target})}{(Accuracy_{Target})}$$



	Acq	Grude	Earn	Grain	Interest	Money-FX	Trade
◆ Naïve Bayes	-0.08	-0.022	-0.029	-0.026	-0.002	-0.321	-0.0475
■ KNN	-0.075	0.035	-0.015	-0.1	0.0274	-0.0156	-0.188
▲ CBA	-0.096	-0.07	-0.097	-0.0264	0.207	-0.219	-0.269
× MCAR	-0.093	0.064	0.01	-0.028	-0.284	-0.198	0.009
✳ BCAR	0.017	-0.064	0.014	0.117	-0.437	0.09	0.058
● MCAR2	0	0	0	0	0	0	0

Figure 5.3: relative BEF for the above scholars against MCAR2

5.7 summery

In this chapter, we examined the effectiveness of AC when applied to TC problem. We applied a newly developed AC algorithm called MCAR2 which employed a newly developed rule pruning method and prediction method. We have selected a large number of text collections from Reuter's corpus to evaluate our developed method. A number of well-known text categorisation algorithms (SVM, KNN, NB) as well as AC methods (CBA, MCAR, BCAR), have been compared with our developed algorithms. The bases of the comparison are the classification accuracy and the break-even-point (BEP) evaluation measures. The empirical studies indicated that the developed algorithm is highly competitive when adopting it to the TC problem.. The revealed results indicate the superiority of MCAR2 when contrasted with other traditional TC classification algorithms such as those of SVM, KNN, and NB in terms of prediction accuracy rate. In conclusion, employing a good pruning procedure in AC as well as TC which keeps only the high quality rules/ features improves the accuracy rate of the constructed classifier; this was proven in the developed algorithm.

CHAPTER SIX

CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

In this thesis, we reviewed the common AC approaches, rule pruning and class assignment approaches. In the light of this review, three main issues in the context of associative classification have been investigated, these are: (1) the exponential growth of the rules generated by AC approaches (2) the bias in class assignment phase when utilising single rule for predicting the class label for a test example and text categorisation problem and (3) adopting AC to TC.

The contribution to the knowledge in this thesis can be summarised as follows: Five pruning methods that consider full and partial coverage with/without class correctness have been developed (Abumansour et al, 2010a, 2010b), new class assignment approach that employs multiple rule for predicting classes for test examples is proposed (Abumansour et al., 2011). Furthermore, a new AC model that employs the best performance pruning method from those proposed in this thesis along with the new class assignment method has been proposed. Lastly, we adapt the proposed AC model to text categorisation problem by integrate a p-re-processing step to the AC model in order to transform the unstructured text data into suitable form to the AC classifier. Following section summarised the thesis contributions:

6.1.1 Adopting Fast intersection approach for rule discovery

Most AC approaches employ Association rule mining for the rule discovery task which often generates enormous number of rules as classification data are usually dense and objects are often highly correlated. Hence, an excessive CPU time is required during the process of discovering the frequent items, generated the rule and learns the classifier which impacts the efficiency. Using smart fast discovery approaches becomes essential.

Most of the current AC approaches in the literature use horizontal data presentation and Apriori method (Agrawal and srikant., 1998) which requires multi passes over the database has been adopted in the rule extraction step. In this thesis we employed a fast intersections method called Tid-list (Zaki and Gouda, 2003) that use vertical data representation which requires single pass over the database. Experiment results against dataset from UCI repository and text corpse revealed that the proposed prediction approaches scores well in efficiency when contrasted with other AC algorithms.

6.1.2 New Rule Pruning methods

Reducing the classifier's size by discarding all redundant and uninteresting rules lead to effective Classifier and accordingly improve the clarification rate. In this thesis, five rule pruning methods have been proposed some of which adopt partial covering and some use full covering and others hybrid: PC, PC^3, FC, FPC and FPCC.

Experiment results revealed that the proposed prediction approaches scores well in term of the time taking in to build the model and classification accuracy

6.1.3 New prediction approach

Most of the current AC algorithms adopt single rule for prediction whereas a fewer adopt multiple rule. Employing single rule in predicting test cases will favour some rules and ignore others which may represent useful knowledge (Li et al., 2001) (Liu et al., 2003) . Employing multiple rules will limits favouring one rule. In this thesis, we proposed a new prediction method that assigns the class label class with highest count of classification rules to test examples.

Experiment results revealed that the proposed prediction approaches scores well when it comes to the classification accuracy

6.1.4 New AC model

In this thesis, a new AC model has been proposed which is an improved version on MCAR (Thabtah et al., 2005). The best performance among the proposed rule pruning methods with respect to efficiency and effectiveness has been selected and employed in MCAR2.

The proposed model has been tested and evaluated through experimentations against data from UCI repository and text corpus, the results revealed that MCAR2 performs better when contrasted with other AC and traditional classification algorithms.

6.1.5 Adapting AC to TC problem

The proposed AC model in this thesis has been adapted to TC problem. The text data has been pre-processed by eliminating the stop words but not stemming. Experimental results against Reuter’s text data revealed MCAR2 can achieve competitive results when contrasting with other algorithms from AC and other classification approaches.

6.2 Future research works

In the following section we discussed a number of future work direction that will be carried out in the near future.

6.2.1 Improve AC effectiveness in terms of Class balancing

Class imbalancing problem is a quite interesting and important issue in data mining context which wasn’t widely investigated by the filed scholars. Class imbalancing has been considered as a crucial problem in machine learning and data mining communities. The problem occurs when there is significantly larger training instances of one class(s) (Majority Class) compared to another class(s) (Minority Class).

Some classification techniques such as decision trees assumes that training cases are consistently scattered among different classes within the dataset while the standard classification approaches tend to ignore or treat those small classes as a noise. This may discard some useful knowledge and decrease accuracy rate.

There is a need to reconsider the issue when having a close look on table 6.1 that depicts the instances distribution in a number of UCI datasets gives an indication that classification accuracy is high for the dataset with balanced or simi-balanced classes; Consider for example the two datasets, Glass and Iris, according to the revealed experimental results in chapter 4, the accuracy rates are 73.18 and 95.133 respectively Glass such as iris, the somehow poor accuracy rate in glass is due a number of reasons including noisy data , missing attributes values and class imbalancing. On the other hand Iris data set scored well with respect to the accuracy rate to the opposite reason in the former, this give an indication that class imbalancing may impact the classification accuracy in some cases. Hence, there is a need for more investigations on the class balancing toward good classification accuracy.

Table 6.1: Some UCI datasets statistics

Dataset	Attributes	Classes	Class Distribution	Tuples
Diabetes	8	2	65% 35% respectively	768
Glass	10	2	76% , 24%	214
Heart	13	2	56%, 44% respectively	270
Iris	4	3	33%, 33%, 33% respectively	150
Labor	16	2	65% 35% respectively	57
Pima	8	2	65% 35% respectively	768
Led7	8	10	10%, 10%, 10%, 8%, 11%, 10%, 11%,9%, 10%, 10%	3200
tic-tac	9	2	65% 35% respectively	958
wine	13	3	33%, 40%, 27% respectively	178
zoo	18	7	41%, 20%, 5%, 13%, 4%, 8%, 10% respectively	101

The issue if class imbalancing was extensively studied by several scholars including (Bermejo et al., 2008) (Phung et al., 2009) (Chin et al., 2012) that the class balancing often improves the classification accuracy. In the near future, we'll investigate and the possibility of employ a class balancing procedure to take place during the rule evaluation step.

6.2.2 Multi-label problem in AC

Most of the current AC approaches are single-label based approaches such as CBA (Liu et al., 1998) CMAR (Li et al., 2001) MCAR (Thabtah et al., 2005). A Single- label classifier considers only the most obvious associated class to a rule and discards all other rules although some of them can be useful for the classifier, this kind of approaches may lead to better accuracy. However, this type of classifiers may not be useful for a number of real life applications where dense datasets are available and there could be multiple classes associated to training object. For instance, medical diagnosis classifications systems, a patient may suffers from food poisoning and cough at the same time. Hence, classifiers that can handle rules with more than one label such as MMAC (Thabtah 2010) HMAC (Sangsuriyun et al., 2010) are required.

To explain the Multi-Label problem further, given two rules such as r_1: $I \rightarrow c_1$ and r_2: $I \rightarrow c_2$, some algorithms consider these rules conflicting (Antonie and Zaine, 2003) consider these two rules conflicting and should be discarded. However, another AC algorithm MMAC (Thabtah et al., 2005) showed by experiments that such rules may be represent knowledge and propose a new technique to deal with such kind of rules. For the above example, MMAC combines and represent these two into the following presentation: r_1: $I \rightarrow c_1 \vee c_2$, an appropriate weight ids assigned for each class according to the frequency for each class in the training dataset.

For real life applications such as text categorisation, medical diagnosis, it’s very important to consider all classes associated with an abject and assign weight to each according to their distribution frequencies in the training. As a result, it is highly needed to develop techniques for multi class and multi label classification system for real world applications that produce the set of all applicable classes that survive a predefined thresholds for each object.

References

[1] Abu-Mansour Hussein, Lee. McCluskey, Fadi thabta and Wael Hadi (2010)Associative Text Categorisation Rules Pruning Method, Proceedings of the Linguistic And Cognitive Approaches To Dialog Agents Symposium, Rafal Rzepka (Ed.), at the AISB 2010 convention,, De Montfort University, Leicester, UK. (pp. 39-44)

[2] Agrawal R., Amielinski T. and Swami A. (1993). Mining association rule between sets of items in large databases. Proceedings of the ACM SIGMOD International Conference on Management of Data, (pp. 207-216). Washington, DC.

[3] Agrawal R. and Srikant R. (1994). Fast algorithms for mining association rule. Proceedings of the 20th International Conference on Very Large Data Bases (pp. 487-499), Santiago, Chile.

[4] Alaa M. El-halees (2006). Arabic Text Classification Using Maximum Entropy .The Islamic University Journal (Series of Natural Studies and Engineering) Vol. 15, No.1, pp 157-167, 2007, ISSN (pp.1726-6807)

[5] Antonie M. and 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.

[6] Antonie M., Zaïane O. R. and Coman A. (2003). Associative Classifiers for Medical Images, Lecture Notes in Artificial Intelligence 2797, Mining Multimedia and Complex Data, (pp. 68-83), Springer-Verlag.

[7] Antonie M. and Zaiane O. (2002). Text Document Categorization by Term Association, Proceedings of the IEEE International Conference on Data Mining (ICDM '2002), (pp.19-26), Maebashi City, Japan.

[8] Baralis E., Chiusano S. and Garza P. (2008). A Lazy Approach to Associative Classification. IEEE Trans. Knowl. Data Eng. 20(2), (pp.156-171.

[9] Baralis E., Chiusano S. and Garza P. (2004). On support thresholds in associative classification. Proceedings of the 2004 ACM Symposium on Applied Computing, (pp. 553-558). Nicosia, Cyprus.

[10] Baralis E. and Paolo Garza.(2012). I-prune: Item Selection for Associative Classification. INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS, VOL. 27(pp. 279–299)

[11] Baralis E. and Paolo Garza, (2002). A Lazy Approach to Pruning Classification Rules. Proceeding ICDM '02 Proceedings of the 2002 IEEE International Conference on Data Mining, (PP. 35-41) Washington, DC, USA.

[12] C. DEISY, M. GOWRI, S. BASKAR, S.M.A. KALAIARASI,N. RAMRAJ (2010). Journal of Engineering Science and Technology Vol. 5, No. 1 (PP. 94 – 107)

[13] Chen J., Yin J., Zhang J., Huang J. (2005) Associative Classification in Text Categorization. ICIC (1) 2005: (pp. 1035-1044).

[14] Cheng-Lung Huang, Mu-Chen Chen , Chieh-Jen Wang. (2007). Credit scoring with a data mining approach based on support vector machines. Expert Systems with Applications 33 (2007) (pp. 847–856)

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

[16] Clark P. and Boswell R. (1991). Rule induction with CN2: Some recent improvements. Proceedings of the Fifth European Working Session on Learning, (pp. 151-163). Berlin, Germany.

[17] CLIFTON PHUA, VINCENT LEE, KATE SMITH1 & ROSS GAYLER. ().A Comprehensive Survey of Data Mining-based Fraud Detection Research. CoRR, 2010.

[18] Diego Kuonen. (2004). A statistical perspective of data mining. CRM Zine (Volume 48)

[19] Duda R. and Hart P. (1973). Pattern classification and scene analysis. John Wiley & son.

[20] Elkourdi M., Bensaid A. and Rachidi T. (2004). Automatic Arabic Document Categorization Based on the Naïve Bayes Algorithm. 20th International Conference on Computational Linguistics. August 28th. Geneva.

[21] Elmasri R. and Navathe S. (2007). Fundamentals of database systems, Fifth Edition, Addison-Wesley.

[22] Fayyad U. and Irani K. (1993). Multi-interval discretisation of continues-valued attributes for classification learning. Proceedings of IJCAI, (pp. 1022-1027). Chambéry, France.

[23] Fayyad U., Piatetsky-Shapiro G., Smith G. and Uthurusamy R. (1998). Advances in knowledge discovery and data mining. AAAI Press.

[24] Freund Y. and Schapire R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci., 55(1):(pp.119–139).

[25] Furnkranz J. and Widmer G. (1994). Incremental reduced error pruning. Proceedings of the Eleventh International Machine Learning Conference, (pp. 70-75). New Brunswick, NJ.

[26] Gan, M., Zhang, M.-Y. &Wang, S.-W. (2005), One extended form for negative association rules and the corresponding mining algorithm, in `Proceedings of 2005 International Conference on Machine Learning and Cybernetics', Vol. 3, (pp. 1716-1721).

[27] Hadi W., Thabtah F., Mousa S. Al Hawari S., Kanaan G. and Ababneh J. (2008a) A Comprehensive Comparative Study using Vector Space Model with K-Nearest Neighbor on Text Categorization Data. Asian Journal of Information Management 2(1): (pp.14-22).

[28] Han J., Pei J. and Yin Y. (2000). Mining frequent patterns without candidate generation. Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, (pp. 1-12). Dallas, Texas.

[29] Harnsamut, N; Natwichai, J; Seisungsittisunti, B (2008). Privacy Preserving of Associative Classification and Heuristic Approach. proceeding of Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2008, ISBN 9780769532639, (pp. 434 – 439).

[30] Joachims T. (1999). Transductive Inference for Text Classification using Support Vector Machines. Proceedings of the International Conference on Machine Learning (ICML), (pp. 200-209).

[31] Joachims T. (1998).Text Categorization with Support Vector Machines: Learning with Many Relevant Features. Proceedings of the European Conference on Machine Learning (ECML), (pp. 173-142), Springer.

[32] John Shafer Rakesh Agrawal Manish Mehta. (1996). SPRINT: A Scalable Parallel Classifier for Data Mining, Proceedings of the 22nd VLDB Conference Mumbai (Bombay), India.

[33] Jyoti Soni, Ujma Ansari, Dipesh Sharma (2011).Predictive Data Mining for Medical Diagnosis. International Journal of Computer Applications, Volume 17– No.8, March 2011.

[34] Kundu G., Islam M., Munir S. and Bari M. (2008). ACN: An Associative Classifier with Negative Rules, Computational Science and Engineering, vol. 0, no. 0, 11th IEEE International Conference on Computational Science and Engineering, (pp. 369-375).

[35] Kononenko, I. (1991). Semi-naive Bayesian classifier. In Proceedings of the 6th European Working Session on Learning, Porto, Portugal (pp. 206–219). Berlin: Springer-Verlag.

[36] Lewis D. (1998a). Reuters 21578 text categorisation test collection. http://www.daviddlewis.com/resources/testcollections/reuters21578.

[37] Lewis D. (1998b). Naive (Bayes) at Forty: the independence assumption in information retrieval (1998b). Proceedings of the 10th European Conference on Machine Learning, (pp. 4-15). Chemnitz, Germany.

[38] Li B., Sugandh N., Garcia E. and Ram A. (2007). Adapting Associative Classification to Text Categorization. In: Proceedings of the 2007 ACM

Symposium on Document Engineering. (ACM DocEng 2007). (pp. 205-207), Winnipeg, Manitoba, Canada.

[39] Li B., Li H., Wu M. and Li P. (2008). Multi-label Classification based on Association Rules with Application to Scene Classification, icycs, 2008 The 9th International Conference for Young Computer Scientists (pp.36-41).

[40] Li and Park (2006). Text categorization based on artificial neural networks. In ICONIP 2006, LNCS 4234. (pp. 302-311).

[41] Li W. (2001). Classification based on multiple association rules. M.Sc. Thesis. Simon Fraser University.

[42] Li W., Han J. and Pei J. (2001). CMAR: Accurate and efficient classification based on multiple-class association rule. Proceedings of the ICDM'01, (pp. 369-376). San Jose, CA.

[43] Lewis, D. D., & Ringuette, M. (1994). Comparison of two learning algorithms for text categorization. In Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval (SDAIR'94).

[44] Li X., Qin D. and Yu C. (2008). ACCF: Associative Classification Based on Closed Frequent Itemsets. FSKD (2) 2008: (pp. 380-384).

[45] Liu B., Hsu W. and Ma Y. (1999). Mining association rules with multiple minimum supports. Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, (pp.337-341). San Diego, California.

[46] Liu B., Hsu W. and Ma Y. (1998). Integrating classification and association rule mining. Proceedings of the KDD, (pp. 80-86). New York, NY.

[47] Liu B., Ma Y. and Wong C-K. (2001). Classification using association rules: weakness and enhancements. In Vipin Kumar, et al, (eds) Data mining for scientific applications, (2001).

[48] Maher Aburrous ,M.A. Hossain , Keshav Dahal, Fadi Thabtah, (2010) .Intelligent phishing detection system for e-banking using fuzzy data mining. Expert Systems with Applications (pp.7913–7921)

[49] Merz C. and Murphy P. (1996). UCI repository of machine learning databases. Irvine, CA, University of California, Department of Information and Computer Science.

[50] M. F. Caropreso, S. Matwin, and F. Sebastiani.(2001). A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization. In Amita G. Chin, editor, Text Databases and Document Management: Theory and Practice, (pp. 78–102). Idea Group Publishing, Hershey, USA.

[51] M. I. Petrovskiy (2003). Outlier Detection Algorithms in Data Mining Systems. Programming and Computer Software, Vol. 29, No. 4, 2003, (pp. 228–237). Translated from Programmirovanie, Vol. 29, No. 4, 2003.

[52] Mitchell M. (1997). Machine Learning, chapter IV, Artificial Neural Networks, (pp. 81-127). WCB/McGraw-Hill, New York, New York.

[53] Niu Q., Xia S. and Zhang L. (2009). Association Classification Based on Compactness of Rules, wkdd, (pp.245-247), Second International Workshop on Knowledge Discovery and Data Mining.

[54] Qian T., Wang Y., Long H. and Feng J. (2005). 2-PS based associative text classification, in: Proceedings of the Seventh International Conference on Data Warehousing and Knowledge Discovery, (pp. 378-387).

[55] Quinlan J. (1998). Data mining tools See5 and C5.0. Technical Report, RuleQuest Research.

[56] Quinlan J. (1993). C4.5: Programs for machine learning. San Mateo, CA: Morgan Kaufmann.

[57] Quinlan J. and Cameron-Jones R. (1993). FOIL: A midterm report. Proceedings of the European Conference on Machine Learning, (pp. 3-20), Vienna, Austria.

[58] Quinlan J. (1986). Induction of decision trees. Machine Learning, 1(1986), (pp.81 – 106).

[59] Rui Xu . (2005).Survey of clustering algorithms. Browse Journals & Magazines > Neural Networks, IEEE Transac ...> Volume: 16 Issue: 3 (pp. 645 - 678).

[60] Sangsuriyun, S. Marukatat, S. ; Waiyamai, K. (2009). Hierarchical Multi-label Associative Classification (HMAC) using negative rules. Cognitive Informatics (ICCI), 2010 9th IEEE International (pp. 919 - 924)

[61] Sebastiani F. (2000). Machine Learning in Automated Text Categorization. ACM Computing Surveys, Vol.34, No.1, (pp. 1-47).

[62] Sebastiani F. (1999). A Tutorial on Automated Text Categorization, Proceedings of the ASAI-99, 1st Argentinian Symposium on Artificial Intelligence, (pp. 7-35).

[63] Snedecor W. and Cochran W. (1989). Statistical Methods, Eighth Edition, Iowa State University Press.

[64] Sparck K. (1972). A statistical interpretation of term specificity and its application in retrieval," Journal of documentation, Vol.28, No.1, (pp. 11-21).

[65] Srinivas.M , K.P.Spreethi, E.V.Prasad, S.Anitha Kumari (2008). MFCC and ARM Algorithms for Text Categorization. In Proceedings ofthe 2008 International Conference on Computing, Communication and Networking (ICCCN 2008)

[66] Tang Z. and Liao Q. (2007). A New Class Based Associative Classification Algorithm. IMECS 2007, (pp. 685-689).

[67] Thabtah F., Eljinini M., Zamzeer M., Hadi W. (2009) Naïve Bayesian based on Chi Square to Categorize Arabic Data. In proceedings of The 11th International Business Information Management Association Conference (IBIMA) Conference on Innovation and Knowledge Management in Twin Track Economies. (pp.930-935).

[68] Thabtah F., Cowling P. and Hamoud S. (2006): Improving Rule Sorting, Predictive Accuracy and Training Time in Associative Classification. Journal of Expert Systems with Applications, Volume 31, Issue 2, (pp. 414-426). Elsevier.

[69] Thabtah, Fadi; Hadi, Wa'el; Abu-Mansour, Hussein; McCluskey, L (2010). Proceeding of 7th International Multi- Conference on Systems, Signals and Devices, 2010, ISBN 9781424475322 (pp. 1 – 6)

Tien Dung Do, Siu Cheung Hui and Alvis C.M. Fong (2006). Associative Feature Selection for Text Mining, International Journal of Information Technology, Vol. 12 No.4 (PP. 59-68)

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

[71] Thabtah, F., Cowling, P., and Peng, Y. (2004) MMAC: A new multi-class, multi-label associative classification approach. Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM ’04), (pp. 217-224). Brighton, UK. (Nominated for the Best paper award).

[72] Tokunaga T., Iwayama M. (1994). Text Categorization Based on Weighted Inverse Document Frequency.1994, in the Special Interest Groups and Information Process Society of Japan (SIG-IPSJ), (pp. 33-39), Tokyo, Japan.

[73] Tien Dung Do, Siu Cheung Hui, and Alvis C.M. Fong.(2005). Artificial Immune System for Associative Classification. ICNC 2005, LNCS 3611, (pp. 849 – 858).

[74] Van Rijsbergan C. (1979). Information retrieval, Buttersmiths, London, 2nd Edition.

[75] Vapnik V. (1995). The Nature of Statistical Learning Theory, chapter 5. Springer-Verlag, New York.

[76] Vyas R., Sharma L., Vyas O. and Schneider S. (2008). Associative Classifiers for Predictive Analytics: Comparative Performance Study, ems, Second UKSIM European Symposium on Computer Modeling and Simulation, 2008 (pp.289-294).

[77] Weka, Data Mining Software. http://www.cs.waikato.ac.nz/ml/weka.

[78] Wiener E., Pedersen O. and Weigend 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).

[79] Witten I. and Frank E. (2000). Data mining: practical machine learning tools and techniques with Java implementations. San Francisco: Morgan Kaufmann.

[80] Wen-Chin Chen, Chiun-Chieh Hsu, Yu-Chun Chu. (2012). Increasing the effectiveness of associative classification in terms of class imbalance by using a novel pruning algorithm, Expert Systems with Applications: An International Journal. Volume 39 Issue 17, (pp. 12841-12850).

[81] Wilbur, J.W., & Sirotkin, K. (1992). The automatic identification of stop words. Journal of Information Science, 18, (pp. 45-55).

[82] Yang Y. (1999). An evaluation of statistical approaches to text categorization, Journal of Information Retrieval, Vol.1 No. 1/2, (pp. 67-88).

[83] Yang Y. and Liu X. (1999). A re-examination of text categorization methods, Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99), (pp. 42-49).

[84] Yang Y. and Pedersen J. (1997). A comparative study on feature selection in text categorization. Proceedings of the Fourteenth International Conference on Machine Learning,(pp. 412– 420). Nashville, TN.

[85] Y.V. Haribhakta and Dr. Parag Kulkarni. (2011). LEARNING CONTEXT FOR TEXT CATEGORIZATION, International Journal of Data Mining & Knowledge Management Process (IJDKP) Vol.1, No.6, November 2011.

[86] Yang Y., Slattery S. and Ghani R. (2002). A study of approaches to hypertext categorization. Journal of Intelligent Information Systems, 18(2002): 149-241.

[87] Yin X. and Han J. (2003). CPAR: Classification based on predictive association rule. Proceedings of the SDM (pp. 369-376). San Francisco, CA.

[88] Yoon Y. and Lee G. (2008). Text Categorization Based on Boosting Association Rules, icsc, 2008 IEEE International Conference on Semantic Computing (PP. pp.136-143).

[89] Yongwook Yoon and Lee, G.G (2008). Text Categorization Based on Boosting Association Rules. Proceeding of 2008 IEEE International Conference on Semantic Computing, 2008, (pp.136 – 143).

[90] Zaïane O. and Antonie A. (2002). Classifying text documents by associating terms with text categories. Proceedings of the Thirteenth Australasian Database Conference (ADC'02), (pp. 215 - 222), Melbourne, Australia.

[91] Zaki M. and Hsiao C-J. (1999). Charm: An efficient Algorithm for closed Association Rules Mining, Technical Report, TR99-10.Computer Science Dept. Rensselaer Polytechnic Institute.

[92] Zaki M. and Gouda K. (2003). Fast vertical mining using diffsets. Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, (pp. 326 – 335). Washington, D.C.

[93] Zaki M., Parthasarathy S., Ogihara M. and Li W. (1997). New algorithms for fast discovery of association rules. Proceedings of the 3rd KDD Conference. (pp. 283-286). Menlo Park, CA.

[94] Zhixin Hao, Xuan Wang, Lin Yao, Yaoyun Zhang. (2009). Improved Classification Based on Predictive Association Rules. Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics San Antonio, TX, USA - October 2009 (pp. 1165-1170)