

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

T04159

Similarity Based Mining for Finding Frequent Itemsets



Developed by

Saif-Ur-Rehman
(176 – FAS/MSCS/04)

Dawlat Khan
(204 – FAS/MSCS/04)

Supervised by

Dr. M. Sikander Hayat Khiyal

Department of Computer Science
International Islamic University, Islamabad.
(2008)

MS
006.322
SAS

- i Data Mining
- ii Database management
- iii Relational databases



MS

Accession No T-4159
SAS

17.07-10

J.S.K.



D.E.
8.12.10

International Islamic University, Islamabad

Date: 26-01-08

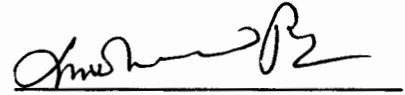
Final Approval

It is certified that we have read the project report titled "*Similarity Based Mining for Finding Frequent Itemsets*" submitted by Mr. Saif-Ur-Rehman Registration No. 176-FAS/MSCS/04 and Mr. Dawlat Khan Registration No. 204-FAS/MSCS/04, in our judgment this project is of sufficient standard to warrant its acceptance by the International Islamic University, Islamabad for the award of MS Degree in Computer Sciences.

Committee

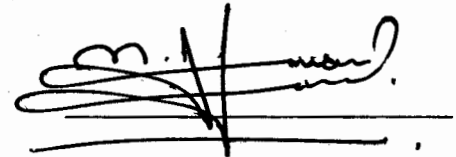
External Examiner

DR Muhammad Riaz
Dean Faculty of sciences
Bahria University Islamabad



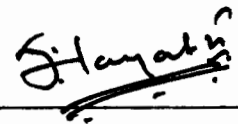
Internal Examiner

Muhammad Imran Saeed
Assistant Professor Deptt of CS
Faculty of Applied Sciences
International Islamic University Islamabad



Supervisor

Professor DR Malik Sikander Hayat Khiyal
Chair Person Deptt of CS / SWE
Fatima Jinnah Women University Rawalpindi



Dedicated to

Our Parents,

Teachers

&

Friends

A dissertation submitted to the
Department of Computer Science,
International Islamic University, Islamabad,
As a partial fulfillment of the requirements
For the award of the degree of
MS in Computer Science

DECLARATION

We, hereby declare that this research, niether as a whole nor as a part, has been copied from any source. It is further declared that we have developed this research entirely based on our personel efforts under the able guidance of our supervisor, Dr. Sikander Hayat Khiyal. If any part of this report is proved to be copied or reported at any stage, we accept the responsibility to face the subsequant consequences. No part of this work inscribed in this report has either been submitted to any other university for the award of degree/qualification.

Saif-Ur-Rehman
176-FAS/MSCS/04

Dawlat Khan
204-FAS/MSCS/04

ACKNOWLEDGEMENTS

All praise to Allah, the Almighty, the most merciful and compassionate, who granted us perseverance to accomplish this research work.

We express our profound gratitude to our kind supervisor **Dr. M.Sikander Hayat Khiyal**. His motivation guided us to this achievement and kept our morale high with his valuable suggestions and personal encouragement. He was readily available for consultations, without which, we never have been able to complete this work.

We are also thankful to **Sir Abdul Salam** (Head Computer Science City University Peshawar) for his inspiring attitude and untiring help during the project efforts. He did a lot of efforts for our success.

We would like also to thank our parents, teachers, friends and all those who helped us and supported us during the project.

Saif-Ur-Rehman
176-FAS/MSCS/04

Dawlat Khan
204-FAS/MSCS/04

Project in Brief

- Project Title:** *Similarity Based Mining for Finding Frequent Itemsets*
- Objective:** The purpose of this research is to study distance based similarity measure for association rule mining. We also succeeded in proving that clustering based measure can also be used for association rule mining. The experimental results showed the effectiveness of the technique.
- Organization:** Department of Computer Science,
International Islamic University, Islamabad.
- Undertaken By:** Saif-Ur-Rehman (176 - FAS/MSCS/04)
Dawlat Khan (204 - FAS/MSCS/04)
- Supervised By:** Professor DR Malik Sikander Hayat Khiyal
Chair Person Deptt of CS / SWE
Fatima Jinnah Women University Rawalpindi
- Language:** C/ C++
- Operating System:** Windows XP
- System Used:** Pentium IV
- Date Started:** February 2006
- Date Completed:** November 2007

Abstract

Finding association rules is a two step process. First step finds frequent itemsets (FIS), which is proved to be NP-Complete. The second step finds association rules from FIS, which is trivial. We present a novel algorithm to find FIS based on clustering measure i.e. jacquard similarity measure. Jacquard similarity measure is based on calculating the distance between itemsets. Our proposed technique makes use of prefix tree as data structure and vertical database layout to cluster related items together. The experimental results have proved that the same FIS can be generated by our technique as compared to other Apriori based algorithms. We can also show that various clustering measures can be applied for association rules mining.

Table of Contents

Chapter No	Contents	Page No
1. INTRODUCTION		
1.1	Emergence of Data Mining	1
1.2	What are Data Mining and Knowledge Discovery.....	2
1.2.1	Data cleaning	4
1.2.2	Data integration	4
1.2.3	Data selection	4
1.2.4	Data transformation	4
1.2.5	Data mining	4
1.2.6	Pattern evaluation	4
1.2.7	Knowledge representation	4
1.3	What kind of Data can be mined.	5
1.3.1	Flat files	6
1.3.2	Relational Databases.....	6
1.3.3	Data Warehouses... ..	7
1.3.4	Transaction Databases	7
1.3.5	Multimedia Databases	7
1.3.6	Spatial Databases	7
1.3.7	Time-Series Databases	7
1.3.8	World Wide Web	7
1.4	Reasons for the growing popularity of Data Mining	8
1.4.1	Growing Data volume	6
1.4.2	Limitation of Human Analysis	8
1.4.3	Low cost of machine learning	9
1.5	How does Data mining work?	9
1.6	Classes	9
1.7	Clusters	10
1.8	Associations	10
2. DATA MINING TECHNIQUES		
2.1	Association Rule.....	11
2.2	Sequence Analysis	11
2.2.1	Algorithms Used in sequential Analysis	13
2.2.1.1	AprioriAll Algorithm	13
2.2.1.2	Apriorisome Algorithm	14
2.3	Classification	14
2.3.1	Classification Methods	15

Chapter No	Contents	Page No
2.3.1.1	Statistical Algorithm	15
2.3.1.2	Neural Network	15
2.3.1.3	Genetic Algorithms	15
2.3.1.4	Nearest neighbor	15
2.3.1.5	Rule induction	16
2.3.1.6	Data visualization	16
2.3.1.7	Naïve-Bayes	16
2.3.1.8	Decision Trees	16
2.3.2	Classification techniques	17
2.3.2.1	Understanding and prediction	18
2.3.3	Algorithms used in classification	19
2.3.3.1	ID3 algorithm	19
2.3.3.2	C4.5 Algorithm	19
2.3.3.3	SQLIQ Algorithm	20
2.4	Clustering	20
2.4.1	Working of Cluster Analysis	21
2.4.2	Uses of Cluster Analysis	21
2.4.3	Further Classification of Cluster Techniques	22
2.4.4	Partitioning Methods	22
2.4.5	Density Based Methods	23
2.4.6	Grid-Based Methods	24
3.	LITERATURE SURVEY	
3.1	Literature Review	25
3.2	Association rule mining	25
3.2.1	Support	26
3.2.2	Confidence	26
3.2.3	Candidate Item set	27
3.3	Association Rule Mining Approaches	27
3.3.1	Apriori Series Approaches.	28
3.3.1.1	AIS Algorithm	28
3.3.1.2	Apriori Algorithm	30
3.3.2	Partition algorithm	32
3.3.3	Eclat Algorithm	33
3.3.3	Sampling approach	34
3.3.4	MaxMiner	35
3.3.5	FP-growth	35
3.3.6	MEDIC	36
3.4	Problem Statement	37

Chapter No	Contents	Page No
4. METHODOLOGY		
4.1. Data structure used for SB-Miner		38
4.2 Node structure used for SB-Miner.....		39
4.3 Database layout		39
4.4 Software architecture		40
4.4.1 Preprocessing module.....		40
4.4.2 FIS finding module using Jacquard similarity (SB-Miner).....		41
5. EXPERIMENTAL RESULTS		
5.1 Introduction.....		45
5.2 Synthetic database generation and results.....		46
5.2 Conclusion.....		47
6. References and bibliography		48
7. Appendix A.....		51

List of Figures and Tables

<u>Figure No</u>	<u>Caption</u>	<u>Page No</u>
1.1	Data Mining and KDD process.....	3
4.1	SE-Tree.....	38
4.2	Alternative representation of SE-Tree.....	39
4.3	Database layouts.....	40
4.4	SB-Miner pseudo code.....	41
5.1	Data Set Conversion Algorithm.....	45
5.2	FIS generation with SB-Miner	46

<u>Table No</u>	<u>Caption</u>	<u>Page No</u>
3.1	AIS Mining Process.....	29
3.2	Apriori Mining Process	31
4.1	Binary Database format	40
5.1	Synthetic binary databases	46
5.2	Max number of FIS (0.5 similarity threshold)	47
5.3	Max number of FIS (0.5 similarity threshold).....	47

CHAPTER 1

Introduction

1. INTRODUCTION

Data mining is the process of extracting interesting (non-trivial, implicit, previously unknown and potentially useful) information or patterns from large information repositories such as: relational database, data warehouses, and XML repository [1].

Data mining refers to extracting or “mining” knowledge from large amounts of data [2]. Mining of gold or sand is referred to as gold mining rather than rock or sand mining. Data mining is often defined as finding hidden information in a database [3]. Similar terms used for data mining are knowledge mining from database, knowledge extraction, data/pattern analysis, data archeology, data dredging, deductive learning, and exploratory data analysis.

1.1 Emergence of Data Mining

The idea of learning from data has been around for a long time. So it is reasonable to ask why the interest in data mining has suddenly become so intense. The principal reason is that the field of Data Base Management has recently become involved Data especially large amount of it, reside in Data Base management system (DBMS). Conventional DBMS are focused on Online Transaction Processing (OLTP) that is the storage and fast retrieval of individual record for purpose of data organization. They are used to keep track of inventory, payroll records, billing records, invoices.

Recently the data base management community has become interested in using DBMS for Decision Support Systems (DSS) allow statistical queries from data collected for OLTP applications. A DSS queries the construction of Data Warehouse. Data Warehouses unify the data scattered throughout the many departments of an organization into a single centralized (usually very large 100 GB) Database with a common format. Sometimes smaller sub Databases are also con-

structured for specialized analysis; these are called Data Marts. Decision Support System are intended for on-line analytical processing (OLAP) and relational On-line analytical processing called ROLAP. ROLAP is intended for multidimensional analysis. ROLAP databases are organized by dimension which is logical grouping by attributes. The conceptual framework is that a data cube which can be viewed as a large high dimensional contingency table.

1.2 What are Data Mining and Knowledge Discovery?

With the enormous amount of data stored in files, databases, and other repositories, it is increasingly important, if not necessary, to develop powerful means for analysis and perhaps interpretation of such data and for the extraction of interesting knowledge that could help in decision-making. Also data mining is known as one of the core processes of Knowledge Discovery in Database (KDD).

Many people take data mining as a synonym for another popular term, Knowledge Discovery in Database (KDD). Alternatively other people treat Data mining as the core process of KDD. The main process of KDD is the data mining process, in these process different algorithms are applied to produce hidden knowledge. After that comes another process called post processing, which evaluates the mining result according to users' requirements and domain knowledge? Regarding the evaluation results, the knowledge can be presented if the result is satisfactory, otherwise we have to run some or all of those processes again until we get the satisfactory result. The actually processes work as follows. First we need to clean and integrate the databases. Since the data source may come from different databases, which may have some inconsistencies and duplication, we must clean the data source by removing those noises or make some compromises. Suppose we have two different databases, different words are used to refer the same thing in their schema. When we try to integrate the two sources we can only choose one of them, if we know that they denote the same thing. And also real world data tend to be incomplete and noisy due to the manual input mistakes. The integrated data sources can

be stored in a database, data warehouse or other repositories. While data mining and knowledge discovery in databases (or KDD) are frequently treated as synonyms, data mining is actually part of the knowledge discovery process. The following Figure 1.1 shows data mining as a step in an iterative knowledge discovery process.

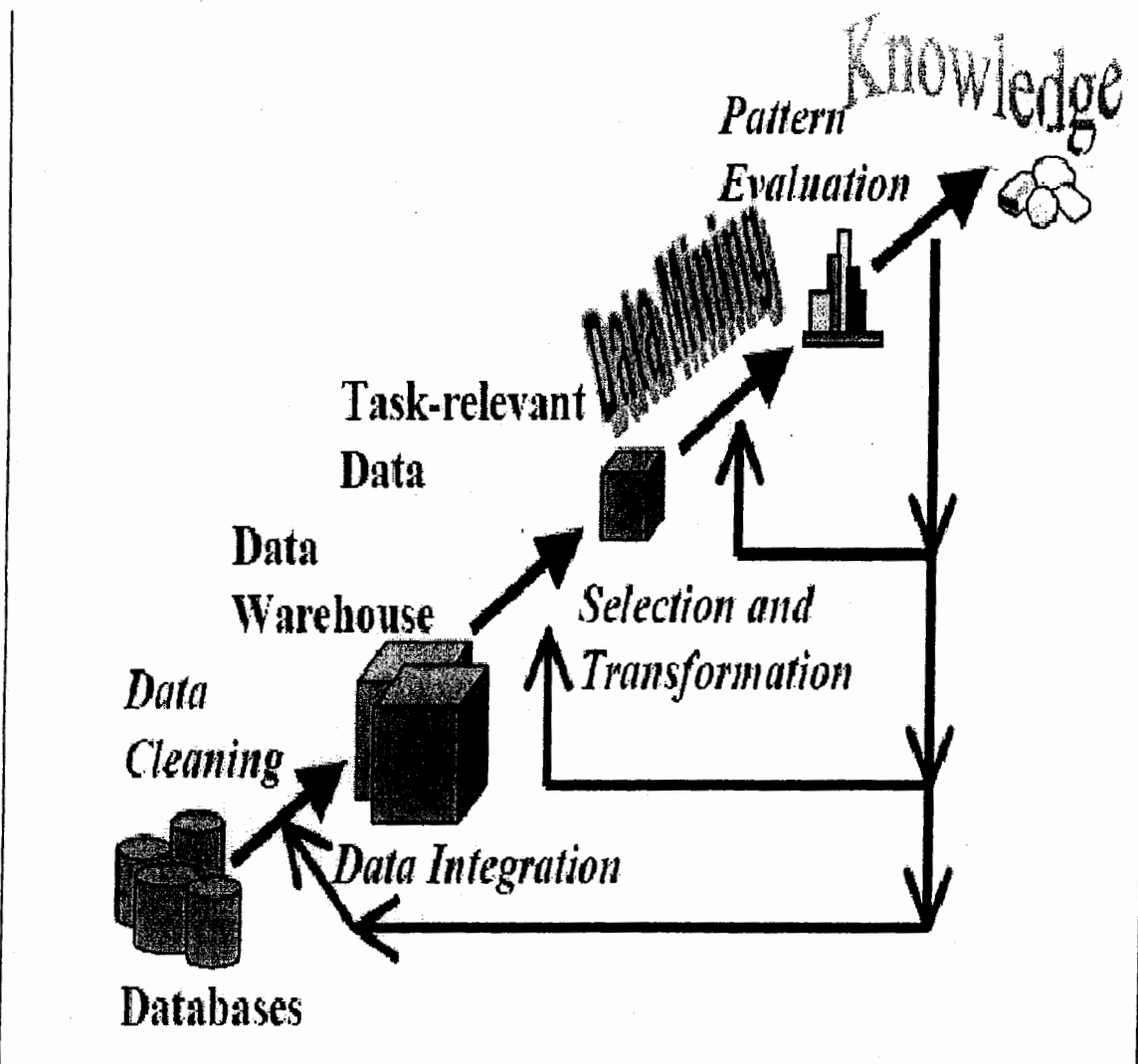


Figure 1.1 Data Mining and KDD process [4]

The Knowledge Discovery in Databases process comprises of a few steps leading from raw data collections to some form of new knowledge. The iterative process consists of the following steps:

1.2.1 Data cleaning: also known as data cleansing, it is a phase in which noise data and irrelevant data are removed from the collection.

1.2.2 Data integration: at this stage, multiple data sources, often heterogeneous, may be combined in a common source.

1.2.3 Data selection: at this step, the data relevant to the analysis is decided on and retrieved from the data collection. This is an important step of the KDD process and sometime more than 50% of the effort in a data mining project is spent on this step [5].

1.2.4 Data transformation: also known as data consolidation, it is a phase in which the selected data is transformed into forms appropriate for the mining procedure.

1.2.5 Data mining: it is the crucial step in which clever techniques are applied to extract patterns potentially useful.

1.2.6 Pattern evaluation: in this step, strictly interesting patterns representing Knowledge is identified based on given measures.

1.2.7 Knowledge representation: is the final phase in which the discovered knowledge is visually represented to the user. This essential step uses visualization techniques to help users understand and interpret the data mining results.

Data cleaning and data integration can be performed together as a pre-processing phase to generate a data warehouse. Data selection and data transformation can also be com-

bined where the consolidation of the data is the result of the selection, or, as for the case of data warehouses, the selection is done on transformed data.

The KDD is an iterative process. Once the discovered knowledge is presented to the user, the evaluation measures can be enhanced, the mining can be further refined, new data can be selected or further transformed, or new data sources can be integrated, in order to get different, more appropriate results. Data mining derives its name from the similarities between searching for valuable information in a large database and mining rocks for a vein of valuable ore. Both imply either sifting through a large amount of material or ingeniously probing the material to exactly pinpoint where the values reside. It is, however, a misnomer, since mining for gold in rocks is usually called “gold mining” and not “rock mining”, thus by analogy, data mining should have been called “knowledge mining” instead. Nevertheless, data mining became the accepted customary term, and very rapidly a trend that even overshadowed more general terms such as knowledge discovery in databases (KDD) that describe a more complete process. Other similar terms referring to data mining are: data dredging, knowledge extraction and pattern discovery.

Data mining involves an integration of techniques from multiple disciplines such as database technology, statistic, machine learning, high-performance computing, pattern recognition, neural networks, data visualization, information retrieval, image and signal processing, and spatial data analysis. There are many applications of data mining.

1.3 What kind of Data can be mined?

In principle, data mining is not specific to one type of media or data. Data mining should be applicable to any kind of information repository. However, algorithms and approaches may differ when applied to different types of data. Indeed, the challenges presented by different types of data vary significantly. Data mining is being put into use and studied for databases, including relational databases, object-relational databases and object-oriented databases, data warehouses, transactional databases, unstructured and semi

structured repositories such as the World Wide Web, advanced databases such as spatial databases, multimedia databases, time-series databases and textual databases, and even flat files. Here are some examples in more detail.

1.3.1 Flat files: Flat files are actually the most common data source for data mining algorithms, especially at the research level. Flat files are simple data files in text or binary format with a structure known by the data mining algorithm to be applied. The data in these files can be transactions, time-series data, and scientific measurements.

1.3.2 Relational Databases: Briefly, a relational database consists of a set of tables containing either values of entity attributes, or values of attributes from entity relationships. Tables have columns and rows, where columns represent attributes and rows represent tuples. A tuple in a relational table corresponds to either an object or a relationship between objects and is identified by a set of attribute values representing a unique key.

1.3.3 Data Warehouses: A data warehouse as a storehouse, is a repository of data collected from multiple data sources (often heterogeneous) and is intended to be used as a whole under the same unified schema. A data warehouse gives the option to analyze data from different sources under the same roof, allows interactive analysis. In other words, data from the different stores would be loaded, cleaned, transformed and integrated together. To facilitate decision making and multi-dimensional views, data warehouses are usually modeled by a multi-dimensional data structure.

1.3.4 Transaction Databases: A transaction database is a set of records representing transactions, each with a time stamp, an identifier and a set of items. Associated with the transaction files could also be descriptive data for the items. Since relational databases do not allow nested tables (i.e. a set as attribute value), transactions are usually stored in flat files or stored in two normalized transaction tables, one for the transactions and one for the transaction items. One typical data mining analysis on such data is the so-called mar-

ket basket analysis or association rules in which associations between items occurring together or in sequence are studied.

Association analysis is useful to find interesting relationship hidden in large data sets. The uncovered relations are represented in the form of association rule or sets of frequent item.

1.3.5 Multimedia Databases: Multimedia databases include video, images, audio and text media. They can be stored on extended object-relational or object-oriented databases, or simply on a file system. Multimedia is characterized by its high dimensionality, which makes data mining even more challenging. Data mining from multimedia repositories may require computer vision, computer graphics, image interpretation, and natural language processing methodologies.

1.3.6 Spatial Databases: Spatial databases are databases that, in addition to usual data, store geographical information like maps, and global or regional positioning. Such spatial databases present new challenges to data mining algorithms.

1.3.7 Time-Series Databases: Time-series databases contain time related data such as stock market data or logged activities. These databases usually have a continuous flow of new data coming in, which sometimes causes the need for a challenging real time analysis. Data mining in such databases commonly includes the study of trends and correlations between evolutions of different variables, as well as the prediction of trends and movements of the variables in time.

1.3.8 World Wide Web: The World Wide Web is the most heterogeneous and dynamic repository available. A very large number of authors and publishers are continuously contributing to its growth and metamorphosis, and a massive number of users are accessing its resources daily. Data in the World Wide Web is organized in inter-connected documents. These documents can be text, audio, video, raw data, and even applications. Con-

ceptually, the World Wide Web is comprised of three major components: The content of the Web, which encompasses documents available; the structure of the Web, which covers the hyperlinks and the relationships between documents; and the usage of the web, describing how and when the resources are accessed. A fourth dimension can be added relating the dynamic nature or evolution of the documents. Data mining in the World Wide Web, or web mining, tries to address all these issues and is often divided into web content mining, web structure mining and web usage mining.

1.4 Reasons for the growing popularity of Data Mining

Day by day the volume of data growing and becoming rocks of data. The aim of Data mining is to search that terabytes of data and find useful information from it. There are many other reasons for growing popularity of data mining such as.

- Growing Data volume
- Limitation of Human Analysis
- Low cost of machine learning

1.4.1 Growing Data volume

The main reason for necessity of automated computer systems for intelligent data analysis is the enormous volume of existing and newly appearing data that require processing. The amount of data accumulated each day by various business, scientific, Medical, World Wide Web and governmental organization around the world is daunting.

1.4.2 Limitation of Human Analysis

The main problem is how to analyze that rocks of raw data and get knowledge from it. Human brain is inadequacy in searching of complex and multifactor dependencies in data. Human experts then see the results of automated tool and extracts knowledge from it.

1.4.3 Low cost of machine learning

One additional benefit of using automated data mining system is that this process has a much lower cost than hiring an army of highly trained professional statisticians. While data mining not eliminate human participation in solving the task completely, it significantly simplifies the job and allow analyst who is not a professional in statistics and programming to manage the process of extracting knowledge.

1.5 How does Data mining work?

While large-scale information technology has been evolving separate transaction and analytical system. data mining provides the link between the two. Data mining software analyzes relationships and patterns in stored transaction data based on open ended user queries. Several types of analytical software are available: statistical, machine learning and neural networks. Generally the following methods are used.

1.6 Classes

Stored data is used to locate data in predetermined groups. Classification maps data into predefined groups or classes. It is often referred to as supervised learning because the classes are determined before examining the data two examples of classification application are determining whether to make a bank loan and identification credit risks. Classification algorithms require that the classes be defined based on data attribute values. They often describe these classes by looking at the characteristics of data already known to belong to the classes. Pattern recognition is a type of classification where an input pattern is classified into one of several classes based on its similarity to these predefined classes. Another example of classes as a restaurant chain could mine customer purchase data to determine when customer visit and what they typically order. This information could be used to increase traffic by having specials.

1.7 Clusters

Data items are grouped according to logical relationship or consumer preferences. For example, data can be mined to identify market segments or consumer affinities. Clustering is similar to classification except that the groups are not predefined, but rather defined by the data alone. Clustering alternatively referred to as unsupervised partitioning or segmenting the data into groups that might or might not be disjointed. The clustering is usually accomplished by determining the similarity among the data on predefined attributes. The most similar data are grouped into clusters.

1.8 Associations

Association is hidden relationship between data items in data sets. This link, analysis alternatively referred to as affinity analysis or association, refer to the data mining task of uncovering relationships among data items in data set. A market basket analysis is the best example of association. Which items are related to each other, another word how much is similarity among data items. The goal of automated tool to determine association rules amongst similar data items.

{Milk} → {Bread}

This shows that milk has strong association with bread because if a customer buys Milk will also buy Bread in a visit.

CHAPTER 2

Data Mining Techniques

2. DATA MINING TECHNIQUES

There have been many advances on researches and developments of data mining and many data mining techniques and system have recently been developed. Different classification schemes can be used to categorize data mining methods and system based on the kind of data bases to be studied, the kind of knowledge to be discovered and the kind of techniques to be utilized.

The data mining categories can be classified into four categories.

- Association Rule
- Sequence Analysis
- Clustering
- Classification

2.1 Association Rule

Association analysis is an unsupervised form of data mining that look for links between records in a data set. Association analysis is some time referred to as 'market basket analysis. Association rule is discus in chapter 3 in detail.

2.2 Sequence Analysis

The input data is set of sequence called data sequences. Each data sequence is an ordered list of transactions (or itemsets) where each transaction is a set of items (literals). Typically there is a transaction-time associated with each transaction. A sequential pattern also consists of a list of sets of items. The problem is to find all sequential patterns with a user-specified minimum support where the support of a sequential pattern is the percentage of data sequential that contain the pattern [26].

An example of such a pattern is that customers typically rent "Stars Wars", then "Empire Strikes Back", and then "Return of the Jedi". Note that these rentals need not be consecutive. Customers who rent some other videos in between also support this sequential pattern. Element of a sequential pattern need not be simple items. "Fitted sheet and flat

sheet and pillow cases” followed by “comforter”, followed by drapes and ruffles is an example of a sequential pattern in which the elements are sets items. This problem was initially motivated by application in the retailing industry, including attached mailing, add-on sales, and customer satisfaction. But the results apply to many scientific and business domains. For instance in the medical domain a data-sequence may correspond to the symptoms or diseases of a patient with a transaction corresponding to the symptoms exhibited or disease diagnosed furring a vast to the doctor. The patterns discovered using this data could be used in disease research to help identify symptoms /diseases that precede certain diseases. Again the user defines the exact value of enough as an input parameter. We define the support of a certain sequences as the fraction of the sequences of the database which contains S. Finding sequential patterns means to discover all the sequences S which have a support superior to a certain threshold.

To give an exact definition of the problems it's necessary to introduce some temporal parameters. These are [26]:

- Sliding time window: when we are looking for a certain itemsets s_i we allow its elements to be spread over different transactions all included in the sliding time window.
- Min_gap: this indicates the minimum time required or two transactions to be considered sequential. If their time indicators differ from a value inferior to Min_gap, the two transactions are considered contemporaneous.
- Max_gap: this indicates the maximum time allows for different transactions to be considered connected and belonging to the same sequence. If two transactions have their time indicators that differ from a value superior to Max_gap, it means that the two transactions have no relationship and must be considered in two separate sequences.

One possible reason for a bank to look for sequential patterns in its database could be to anticipate the moves of its customers. For example, it could be discovered that unhappy customers before closing their account, have similar behaviors. This could allow the bank

to understand when a customer is not satisfied with the service and to take precautions not to lose him or her. Other examples of application are again in the field of medicine. Mining sequential patterns could help to connect the treatments with the course of the disease in a certain patient.

2.2.1 Algorithms Used in sequential Analysis:

Various groups working in this field have suggested algorithms for mining sequential patterns. Listed below are two algorithms proposed by IBM's Quest data team.

There are two families of algorithms—count-all and count-some. The count-all algorithms count all the sequences, including non-maximal sequences. The non-maximal sequences must then be pruned out (in the maximal phase). AprioriAll listed below is a count-all algorithm, based on the Apriori algorithm. The intuition behind these algorithms is that since we are only interested in maximal sequences, we can avoid counting sequences, which are contained in a longer sequence if we first count longer sequences. However, we have to be careful not to count a lot of longer sequences that do not have minimum support. Otherwise, the time by not counting sequences contained in a longer sequence may be less than the time wasted counting sequences without minimum support that would never have been counted because their subsequences were not large.

2.2.1.1 AprioriAll Algorithm:

The algorithm is given below [27]. In each pass, we use the large sequences from the previous pass to generate the candidate sequences and then measure their support by making a pass over the database. At the end of the pass, the support of the candidates is used to determine the large sequences. In the first pass, output of the itemset phase is used to initialize the set of large 1-sequences. The candidates are stored in hash-tree to quickly find all candidates contained in a customer.

L_1 = large 1-sequences: // result of itemset phase

For ($k=2$; $L_{k-1} \neq \emptyset$; $k++$) do

Begin

C_k = new candidates generated from L_{k-1} (see below)

For each customer-sequence c in the database do

Increment the count of all candidates in c_k that are contained in c .

L_k = candidates in c_k with minimum support.

End

Answer = Maximal sequence in $\bigcup_k L_k$;

2.2.1.2 Apriorisome Algorithm:

In this algorithm [27], we only count sequence of certain lengths. For example, we might count sequences of length 1, 2, 4 and 6 in the forward phase and count sequences of length 3 and 5 in the backward phase. The function `next` takes as parameter the length of sequence counted in the last pass and returns the length of sequences to be counted in the next pass. Thus, this function determine exactly which sequence are counted, and balance the tradeoff between the time wasted in counting non-maximal sequence versus counting extensions of small candidate sequences. One extreme is `next(k) = k+1` (k is the length for which candidates were counted last); when all non-maximal sequences are counted but no extension of small candidate sequences are counted. In this case, Apriori some degenerates into AprioriAll. The other extreme is a function like `next(k) = 100*k`, when almost no non-maximal large sequence is counted, but lot of extensions of small candidates are counted.

2.3 Classification

Classification identifies a specific group or class to which an item belongs. A prediction based on a classification model will, therefore, be a discrete outcome, Identifying customer as a responder or non-responder, or a patient as having a high or low risk of heart failure.

Classification is the operation most commonly supported by commercial data mining tools. It is an operation that enables organizations to discover patterns in large or complex data sets in order to solve specific business problems.

Classification is the process of sub-dividing a data set with regard to a number of specific outcomes. For example, we might want to classify our customers into 'high' and 'low' categories with regard to credit risk. The category or 'class' into which each customer is placed is the 'outcome' of our classification.

A crude method would be to classify customers by whether their income is above or below a certain amount. A slightly more subtle approach tries to find a linear relationship between two different factors such as income and age to divide a data set into two groups. Real world classification problems usually involve many more dimensions and therefore require a much complex delimitation between different classes.

2.3.1 Classification Methods

There have been many data classification methods studied:

2.3.1.1 Statistical Algorithm: Statistical analysis system such as SAS and SPSS has been used by analysis to detect unusual patterns and explain patterns using statistical models such as linear models. Such systems have their place and will continue to be used.

2.3.1.2 Neural Network: Artificial neural networks mimic the pattern-finding capacity of the human brain and hence some researchers have suggested applying Neural Network Algorithms to pattern mapping. Neural networks have been applied successfully in a few applications that involve classification.

2.3.1.3 Genetic Algorithms: Optimization techniques that use processes such as genetic combination, mutation, and natural selection in a design based on the concepts of natural evolution.

2.3.1.4 Nearest neighbor: Nearest Neighbor is a predictive technique suitable for classification models. Unlike other predictive algorithms, the training data is not

scanned or processed to create the model. Instead, the training is the model. When a new case or instance is present to the model, the algorithm looks at all the data to find a subset of cases that are most similar to it and use them to predict the outcome. As the term nearest implies, k-NN is based on a concept of distance, and this requires a metric to determine distances. All metrics must result in a specific number for comparison purposes. Whatever metric is used is both arbitrary and extremely important. It is arbitrary because there is no present definition of what constitutes a “good” metric. It is important because the choice of a metric greatly affects the predictions. Different metrics, used on the same training data, can result in completely different predictions. This means that a business expert is needed to help determine a good metric.

2.3.1.5 Rule induction: The extraction of useful if-then rules from data based on statistical significances.

2.3.1.6 Data visualization: The visual interpretation of complex relationships in multidimensional data.

2.3.1.7 Naïve-Bayes: Naïve-Bayes is a classification technique that is both predictive and descriptive. It analysis the relationship between each independent variables and the dependent variables to derive a conditional probability for each relationship. Naïve-Bayes requires only one pass through to the training set to generate a classification model. This makes it the most efficient data mining technique. However, Naïve-Bayes does not handle continuous data, so any independent or dependent variables that contain continuous data values must binned or bracketed. For instance, if one of the independent variable is age, the value must be transformed from the specific value into ranger such as “less then 20 years,” “21 to 30 years,” “31 to 40 years” and so on.

2.3.1.8 Decision Trees: Decision Trees is one of the most common data mining techniques and are by for the most popular in tools aimed at the business users. They are easy to set up their results are understandable by an end-user. They can address a wide range of classification problems. They are robust in the face of dif-

ferent data distributions and formats, and they are effective in analyzing large number of fields. A decision tree algorithm works by splitting a data set in order to build a model that successfully classifies each record in terms of a target field or variable. An example is decision tree, which classifies a data set according to whether customers did or did not buy a particular product. The most common types of decision tree algorithm are CHAID, CART and C4.5. CHAID (Chi-square automatic interaction detection) and CART (Classification and Regression Trees) were developed by statisticians. CHAID can produce tree with multiple sub-nodes for each split. CART requires less data preparation than CHAID, but produces only two-way splits. C4.5 comes from the world of machine learning, and is based on information theory. In order to generate a decision tree from the training set of data we need to split the data into progressively smaller subsets. Each iteration considers the data in only one node. The first iteration considers the root node that contains all the data. Subsequent iterations work on derivative nodes that will contain subsets of the data.

2.3.2 Classification techniques

How the data-mining tool analyses this data, and the type of information it provides on the techniques it uses. The most common techniques for classification is decision trees and neural networks. If a decision tree is used, it will provide a sort of branching condition that successively split the customers into groups defined by the values in the independent variables. The aim is to be able to produce a set of rules or a model of some sort that can identify a high percentage of responders. A decision tree may formulate a condition such as:

In contrast, a neural network identifies which class a customer belongs to, but cannot tell you why. The factors that determine its classifications are not available for analysis, but remain implicit in the network itself. Another set of techniques used for classification are k-nearest neighbor's algorithms.

2.3.2.1 Understanding and prediction

Sophisticated classification techniques enable us to discover new patterns in large and complex data sets. Classification is therefore a powerful aid to understanding a particular, whether it is response rate to a direct mailing campaign or the influence of various factors on the likelihood of a patient recovering from cancer.

In some instances, improved understanding is sufficient. It may suggest new initiatives and provide information that improves future decision-making. However, in many instances the reason for developing an accurate classification model is to improve our capability for prediction.

For example, we know that historically 60 percent of customers, who are male, married and have incomes over £60,000 responded to a promotion (compared with only three percent of all targeted customers). There is therefore a better than average chance that new customers who fit this profile will also be interested in our product. In practice, predictor variables, thus providing a much more complex relationship involving numerous predictor variables, thus provide a much finer segmentation of customers.

A classification model is said to be 'trained' on historical data, for which the outcome is known for each record. It is then applied to a new, unclassified data set in order to predict the outcome for each record.

There are important difference between classifying data in order to understand the behavior of existing customers and using that classification to product future behavior. For a historical data set, it is often possible to produce a set of rules or a mathematical function that classifies every record accurately. For example, if you keep refining your rules you end up with a rule for each individual of the form.

100 percent of customers called smith who live at 28 Arcades Street responds to our offer.

Such a rule is of little use for predicting the classification of a new customer. In this case, the model is said to be 'over-trained' or 'over fitted' to the historical data set. Build-

ing a good predictive model requires that over fitting be avoided by testing and tuning a model to ensure that it can be 'generalized' to new data.

2.3.3 Algorithms used in classification:

There are many different algorithm used for the classification. Some of them are described in brief below.

2.3.3.1 ID3 algorithm:

The ID3 algorithm (Quinlan86) is a decision tree building algorithm which determines the classification of objects by testing the values of their properties [6]. It builds the tree in a top down fashion starting from a set of object and a specification of properties. At each node of tree a property is tested and the results used to partition the object set. The process is recursively done till the set in a given sub tree is homogeneous with respect to the classification criteria in other words it contains objects belonging to the same category. This then becomes a leaf node. At each node the property to test is chosen based on information theoretic criteria that seek to maximize information gain as minimize entropy. In simpler terms that property is testes, which divides the candidate set in the most homogenous subsets.

2.3.3.2 C4.5 Algorithm:

This algorithm was proposed by Quinlan (1993) [7]. The C4.5 algorithm generates a classification decision tree for the given data set by recursive partitioning of data. The decision is grown using Depth -first strategy. The algorithm all the possible tests that can split the data set and selects a test that gives the best information gain. For each discrete attribute one test with outcomes as many as the number of distinct values of the attribute is considered. For each continuous attribute binary tests involving every distinct values of the attribute are considers. in order to gather the entropy gain of all these binary tests efficiently the training data set belonging to the node in consideration is sorted for the values of the continuous attribute and the entropy gains of the binary cut based on each distinct

values are calculated in one scan of the data. This process is repeated for each continuous attribute.

2.3.3.3 SQLIQ Algorithm:

SQLIQ (Supervised learning in quest) developed by IBM's quest project team is a decision tree classifier designed to classify large training data [8]. It uses a pre-sorting technique in the tree-growth phase. This helps avoid costly sorting at each node. SQLIQ keeps a separate sorted list for each continuous attribute and a separate list called class list. An entry in the class list corresponds to a data item and has a class label and name of the node it belongs in the decision tree. An entry in the sorted attribute list has an attribute value and the index of data item in the class list. SQLIQ grows the decision tree in breadth-first manner. For each attribute it scans the corresponding sorted list and calculate entropy values of each distinct values of all the nodes in the frontier of the decision tree simultaneously. After the entropy values have been calculated for each attribute one attribute is chosen for split for each node in the current frontier and they are expended to have a new frontier. Then one more scan of the sorted attribute list is performed to update the class list for the new nodes. While SQLIQ handles disk-resident data that are too large to fit in memory it still requires some information to stay memory-tree based data structure.

2.4 Clustering

Clustering is an unsupervised operation. It is used where you wish to find groupings of similar records in your data without any preconditions as to what that similarity may involve. Clustering is used to identify interesting groups in a customer base that may not have been recognized before. For example, it can be used to identify similarities in customer's telephone usage. In order to devise and market new call services.

Clustering is usually achieved using statistical methods, such as a k-means algorithm. Or a special form of neural network called a Kohonen feature map network. Whichever

method is used, the basic operation is the same. Each record is compared with a set of existing clusters. Which are defined by their 'center'? A record is assigned to the cluster it is nearest to, and this in turn changes the value that defines that cluster. Multiple passes are made through a data set to re-assign record and adjust the cluster centers until an optimum solution is found.

2.4.1 Working of Cluster Analysis

Cluster analysis is the process of identifying the relationships that exist between items on the basis of their similarity and dissimilarity. Unlike classification, clustering does not require a target variable to be identified beforehand. A clustering algorithm takes an unbiased look at the potential groupings within a data set and attempts to derive an optimum delineation of items on the basis of those groups.

Clusters are typically base around a "center" or average value. How centers are initially defined and adjusted varies between algorithms. One method is to start with a random set of centers, which are then adjusted, added to and removed as the analyses progresses.

To identify items that belong to a cluster, some measure must be used that gauges the similarity between items within a cluster and their dissimilarity to items in other clusters. The similarity and dissimilarity between items is typically measured as their distance from each other and from the cluster centers within a multi-dimensional space, where each dimension represents one of the variables being compared.

2.4.2 Uses of Cluster Analysis

- Scalability
- Ability to deal with different types of attributes
- Discovery of clusters with arbitrary shape (not only spherical clusters)
- Minimal requirements for domain knowledge to determine input parameters (such as # of clusters)
- Able to deal with noise and outliers

- Insensitive to order of input records
- High dimensionality (especially very sparse and highly skewed data)
- Incorporation of user-specified constraints
- Interpretability and usability (close to semantics)

2.4.3 Further Classification of Cluster Techniques

Clusters can be further divided into four categories.

- *Partitioning Method*: Construct various partitions and then evaluate them by some criterion.
- *Hierarchical Method*: Create a hierarchical decomposition (agglomerative or divisive) of the set of data (or objects) using some criterion.
- *Density based Method*: based on connectivity and density functions.
- *Grid Based Method*: based on a multiple-level granularity structure.

2.4.4 Partitioning Methods

Construct a partition of a database D of n objects into set of k cluster given a k ; find a partition of k clusters that optimizes the chosen partitioning criterion.

It consists of these algorithms:

- K-means
- K-medoid
- K-Plane
- CLARANS (A Clustering Algorithm based on Randomized Search) (Ng and Han 94) it is improved version of k-medoid.
- K-modes: extension of K-mean
- Bisecting K-means: better results than k-mean

- PAM (Kaufman and Rousseeuw, 1987),: starts from an initial set of medoid and iteratively replaces one of the medoid by one of the non-medoid if it improves the total distance of the resulting clustering
- CLARA (Kaufmann and Rousseeuw in 1990): It draws multiple samples of the data set, applies PAM on each sample, and gives the best clustering as the output number of clusters k as an input, but needs a termination.

It consists of these algorithms:

- AGNES (Agglomerative Nesting)" [6] introduced in kaufmann and rousseeuw (1990). In the k -mean approach to clustering we start out with a fixed number of clusters and gather all records into them. There is another class of methods that work by agglomeration. In these methods we start out with each data point forming its own cluster and gradually merge clusters until all points have been gathered together in one big cluster. Towards the beginning of the process the clusters are very are very small and very pure- the members of each clusters are few but very closely related. Towards the end of the process the clusters are large and less well defined. The entire history is preserved so you can choose the level of clustering that works best for your application.
- DIANA (Divisive Analysis) introduced in kaufinana and rousseeuw (1990) [8].
- BIRCH (1996): uses CF-tree and incrementally adjust the quality of sub-clusters.
- CURE (1998): select well –scattered (representative) points from the cluster and then shrinks them towards the center of the cluster by a specified fraction.
- CHAMELEON (1999): hierarchical clustering using dynamic modeling.

2.4.5 Density Based Methods:

The can discover clusters with arbitrary shape and they do not need to present number of cluster. It consists of these algorithms:

- DBScan
- TERN

- EM(Expectation Maximization)

2.4.6 Grid-Based Methods:

It first quantizes the clustering space into finite number of cells and then performs clustering on the girded cells.

It consists of the following algorithm:

- Wave Cluster
- Clique

CHAPTER 3

Literature Survey

3. LITERATURE SURVEY

3.1 Literature Review

Data mining is used to discover patterns and relationships in data, with an emphasis on large observational data bases. Data mining is performed in two steps. First existing data is transformed into pattern. Patterns are data or items having same properties or rules. Second Data mining tools are applied on these patterns and knowledge is extracted from it. Knowledge is information on which decisions are being made. There are several fields on which data mining is used including data base management system, Artificial intelligence, Machine learning, Pattern Recognition and Data Visualization. A statistical perspective is this that it can be viewed as computer automated exploratory data analysis of large complex data sets. In spite of some what exaggerated hype, this field is having a major impact in business industry and science. It also affords enormous research opportunities for new methodological development.

3.2 ASSOCIATION RULE MINING

Association rule mining explores for interesting relationships among items in a given data set. In other words Association rule mining helps in finding interesting association relationships among large set of data items. The discovery of such associations rule can help develop strategies to predict.

Association analysis is used for market basket or transaction data analysis. Market basket process finds associations between the different items. An objective of association rule mining is to develop a systematic method using the given data set and finds relationships between the different items. Association is a rule, which implies certain association relationships among set of objects such as occur together or one implies the other[9]. Goal of association rule is finding associations among items from a set of transactions which contain a set of items. Main problem of Association Rule inductions is that there are so many possible rules.

A typical association rule mining algorithm has the following components:

- Structure (used for implementation of the pattern)
- Score function (simply binary function, support and confidence)
- Search method (systematic search methods)
- Association rule mining is used to find frequent Itemset. A support and confidence of items are calculated.

Efficient algorithms are needed which restrict the search space to minimize the algorithmic complexity and interesting rules must be picked from the set of generated rules. The search space for enumeration of all frequent item sets is 2^m , where m represents the number of items. The rule generation step's complexity is $O(r \cdot 2^l)$, where r is the number of frequent item sets, and l is the longest frequent item set [10]. Mining association rules can be divided into two sub-problems:

- Find all frequent itemsets which have the support greater than the minimum pre-determined min_sup in the transaction database.
- Generate all association rules $X \Rightarrow Y$, which have confidence greater than
- Predetermined min_conf .

3.2.1 Support

Support(s) of an association rule is defined as the percentage/fraction of records that contain $X \cup Y$ to the total number of records in the database. Support can be calculated by applying the following formula.

$$\text{Support (XY)} = \frac{\text{Support count of XY}}{\text{Total number of transaction in D}}$$

3.2.2 Confidence

Confidence of an association rule is defined as the percentage/fraction of the Number of transactions that contain XUY to the total number of records that Contain X , where if

the percentage exceeds the threshold of confidence an interesting association rules XUY can be generated.

$$\text{Confidence}(X/Y) = \frac{\text{Support}(XY)}{\text{Support}(X)}$$

Confidence is a measure of strength of the association rules, suppose the confidence of the association rule $X \Rightarrow Y$ is 80%, it means that 80% of the transactions that contain X also contain Y together, similarly to ensure the interestingness of the rules specified minimum confidence is also pre-defined by users.

3.2.3 Candidate Item set

The first sub problem described in section 3.2 mining associations rules can be further divided into two sub problems: candidate large Itemsets generation process and frequent Itemsets generation Process. We call those Itemsets whose support exceed the support threshold as large or frequent Itemsets, those Itemsets that are expected or have the hope to be large or frequent are called candidate Itemsets. Most of the algorithms of mining association rules we surveyed are quite similar, the difference is the extend to which certain improvements have been made, so only some of the milestones of association rule mining algorithms will be introduced. There are different algorithms used for finding frequent itemset. Some of the approaches are as follows:

3.3 Association Rule Mining Approaches

Association rule mining is a well explored research area, we will only introduce some basic and classic approaches for association rule mining. As stated before, the second sub problem of ARM is straightforward, most of those approaches focus on the First sub problem. The first sub problem can be further divided into two sub problems: candidate large itemsets generation process and frequent itemsets generation Process. We will introduce some approaches for association rule mining. In order to make it easier for us to compare algorithms we use the same transaction database, a transaction database from a

supermarket, to explain how those algorithms work. This database records the purchasing attributes of its customers. Suppose during the preprocess all those attributes that are not relevant or useful to our mining task are pruned, only those useful attributes are left ready for mining as shown in Table 3.1 (a).

3.3.1 Apriori Series Approaches

In this section we will discuss all those approaches which are based on support count.

3.3.1.1 AIS Algorithm.

The AIS (Agrawal, Imielinski, and Swami) algorithm was the first algorithm proposed for mining association rule in [11]. It focuses on improving the quality of databases together with necessary functionality to process decision support queries. In this algorithm only one item consequent association rules are generated, which means that the consequent of those rules only contain one item, for example we only generate rules like $X \cap Y \Rightarrow Z$ but not those rules as $X \Rightarrow Y \cap Z$

The databases were scanned many times to get the frequent itemsets in AIS. During the first pass over the database, the support count of each individual item was accumulated as shown in Table 3.1 (b), suppose the minimal support threshold is 30%, large one items were generated in Table 3.1 (c). According to minimal support those items whose support counts are less than 3 (I_4 and I_6) are eliminated from the list of frequent items. With those frequent 1 items, candidate 2-itemsets are generated by extending those frequent items with other items in the same transaction. To avoid generating the same itemsets repeatedly the items were ordered, candidate itemsets are generated by joining the large items in previous pass and another item in the transaction, which appears later than the last item in the frequent itemsets. For example, based on transaction T100 I_1, I_2, I_5 according to this specific order we generate candidate 2-itemsets by extending I_1 with only $I_2; I_5$, similarly I_2 is extended with I_5 . The result is shown in Table 3.1(d). During the second pass over the database, the support count of those candidate 2-itemsets are accumulated and checked against the support threshold. Similarly those candidate (k+1)-itemsets are generated by extending frequent k-itemsets with items in the same transaction.

Table 3.1 AIS Mining Process

TID	List of Items
T100	I ₁ , I ₂ , I ₅
T200	I ₂ , I ₄
T300	I ₂ , I ₃
T400	I ₁ , I ₃
T500	I ₁ , I ₂ , I ₄
T600	I ₂ , I ₃
T700	I ₁ , I ₃
T800	I ₁ , I ₂ , I ₃ , I ₅
T900	I ₁ , I ₂ , I ₃
T1000	I ₁ , I ₂ , I ₅ , I ₆

(a) Original Database

Items	Count number
I ₁	7
I ₂	8
I ₃	6
I ₄	2
I ₅	3
I ₆	1

(b) C₁

{I₁, I₂, I₅}

{I₁, I₂}

{I₂, I₃}

Large 1 Items
I ₁
I ₂
I ₃
I ₅

{I₁, I₂, I₃, I₅}

(c) L₁

T-4159

Items	Count number
I ₁ , I ₂	5
I ₁ , I ₅	3
I ₂ , I ₅	3
I ₂ , I ₄	2
I ₂ , I ₃	4
I ₁ , I ₄	1
...	...

(d) C₂

Large 2 Items
I ₁ , I ₂
I ₁ , I ₅
I ₂ , I ₅
I ₂ , I ₃
I ₁ , I ₃

(e) L₂

Items	Count number
I ₁ , I ₂ , I ₅	3
I ₁ , I ₂ , I ₄	1
I ₁ , I ₂ , I ₆	2
I ₁ , I ₂ , I ₄	1
I ₂ , I ₃ , I ₅	1
I ₁ , I ₃ , I ₅	1
.....	...

(f) C₃

All those candidate itemsets generation and frequent itemsets generation process iterate until any one of them becomes empty. The result frequent itemsets includes only one large 3-itemsets I_1 , I_2 and I_5 . To make this algorithm more efficient, an estimation method was introduced to prune those itemsets candidates that have no hope to be large, consequently the unnecessary effort of counting those itemsets can be avoided. Since all the candidate itemsets and frequent itemsets are assumed to be stored in the main memory, memory management is also proposed for AIS when memory is not enough. One approach is to delete candidate itemsets that have never been extended. Another approach is to delete candidate itemsets that have maximal number of items and their siblings, and store this parent itemsets in the disk as a seed for the next pass. The detail examples are available in [11].

The main drawback of the AIS algorithm is too many candidate itemsets that finally turned out to be small are generated, which requires more space and wastes much effort that turned out to be useless. At the same time this algorithm requires too many passes over the whole database.

3.3.1.2 Apriori Algorithm

Apriori is a great improvement in the history of association rule mining, Apriori algorithm was first proposed by Agrawal in [12]. The AIS is just a straightforward approach that requires many passes over the database, generating many candidate itemsets and storing counters of each candidate while most of them turn out to be not frequent. Apriori is more efficient during the candidate generation process for two reasons; Apriori employs a different candidate's generation method and a new pruning technique.

Table 3.2 Apriori Mining Process

Items	Count number
I ₁	7
I ₂	8
I ₃	6
I ₄	2
I ₅	3
I ₆	1

(a) C₁

Large 1 Items
I ₁
I ₂
I ₃
I ₄
I ₅

(b) L₁

Items	Count number
I ₁ , I ₂	5
I ₁ , I ₃	4
I ₁ , I ₅	3
I ₂ , I ₃	4
I ₃ , I ₅	3
I ₂ , I ₅	1

(c) C₂

Large 2 Items
I ₁ , I ₂
I ₁ , I ₅
I ₂ , I ₅
I ₂ , I ₃
I ₁ , I ₃

(d) L₂

Items	Count number
I ₁ , I ₂ , I ₅	3
I ₁ , I ₂ , I ₃	2

(e) C₃

There are two processes to find out all the large itemsets from the database in Apriori algorithm. First the candidate itemsets are generated, and then the database is scanned to check the actual support count of the corresponding itemsets. During the first scanning of the database the support count of each item is calculated and the large 1-itemsets are generated by pruning those itemsets whose supports are below the pre-defined threshold as shown in Table 3.2(a) and (b). In each pass only those candidate itemsets that include the same specified number of items are generated and checked. The candidate k-itemsets are generated after the (k-1) the passes over the database by joining the frequent k-1 - itemsets. All the candidate k-itemsets are pruned by check their sub (k-1)-itemsets, if any of its sub (k-1)-itemsets is not in the list of frequent (k-1)-itemsets, this k-itemsets candi-

date is pruned out because it has no hope to be frequent according the Apriori property. The Apriori property says that every sub (k-1)-itemsets of the frequent k-itemsets must be frequent. Let us take the generation of candidate 3-itemsets as an example. First all the candidate itemsets are generated by joining frequent 2-itemsets, which include (I_1, I_2, I_5) , (I_1, I_2, I_3) , (I_2, I_3, I_5) , (I_1, I_3, I_5) . Those itemsets are then checked for their sub itemsets, since (I_3, I_5) is not frequent 2-itemsets, the last two 3-itemsets are eliminated from the list of candidate 3-itemsets as shown in Table 3.2 (e). All those processes are executed iteratively to find all frequent itemsets until the candidates' itemsets or the frequent itemsets become empty. The result is the same as the AIS algorithm. The algorithm is shown in Figure 3. In the process of finding frequent itemsets, Apriori avoids the effort wastage of counting the candidate itemsets that are known to be infrequent. The candidates are generated by joining among the frequent itemsets level-wisely, also candidate are pruned according the Apriori property. As a result the number of remaining candidate itemsets ready for further support checking becomes much smaller, which dramatically reduces the computation, I/O cost and memory requirement. Table 3.2 shows the process of Apriori algorithm, by compare Table 3.1 and Table 3.2 we can see the numbers of candidates changed dramatically. Detail of the Apriori-gen and Generate Rules functions were elaborated in [12]. Apriori algorithm still inherits the drawback of scanning the whole data bases many times. Based on Apriori algorithm, many new algorithms were designed with some modifications or improvements.

3.3.2 Partition algorithm

Partition is a different approach from all the others. Algorithm use set intersection operation instead of counting [13]. Supports of an itemsets evaluated with its subsets. Partition algorithm combines the Apriori with set operation (intersection).

Partition algorithm stores the complete database which includes the transaction sets, into main memory. For huge databases, this could be impossible. Nevertheless, the partition algorithm partitions database into several chunks to solve the problem that, is the main memory limitations [14].

Partition algorithm is used to find FIS, which logically divides the horizontal database into a number of non-overlapping partitions. Each partition is read, and vertical tid-lists are formed for each item, i.e. list of a tid's where the item appears. Then, all locally frequent item sets are generated via tid-list intersections. All locally frequent item sets are merged and a second pass is made through all the partitions. The database is again converted to the vertical layout and the global counts of all the chosen item sets are obtained. The size of a partition is chosen, so that it can be accommodated in main-memory. Partition, thus makes only two database scans. The key observation used is that a globally frequent itemsets in at least one partition must be locally frequent in at least one partition. This approach also uses support measure. Partition algorithms improve the performance with using important principles "any globally frequent itemset must appear as a locally frequent itemset in at least one of the partitions of the database" [15].

The partition algorithm merges all local frequent itemsets of every part. If an item is frequent in the entire database, it must be relatively frequent in one of the parts. First, every part is read into main memory using the vertical database layout next, frequent itemsets are found into each partitions then, support of every itemsets are computed. Finally, global itemset frequency is determined using the intersection operation.

If the database is heterogeneous then it decreases the performance of the algorithm. Because, there can be too many local frequent itemsets. On the other hand, if the entire database fits into main memory, then divides the transactions into several partitions not necessary. The penalty in partition/sampling are discussed and concluded in [16] by Frans et al (2002) that candidate set derived is necessarily a superset of the actual set of frequent item sets and may contain many false positives.

3.3.3 Eclat Algorithm

Eclat is an extension of Apriori algorithm and presents different design approach for association rule mining. Eclat uses the vertical database layout and uses the intersection based approach to compute the support of an itemset [17].

Eclat constructs candidate itemsets using the join step from Apriori. Eclat differs from the Apriori algorithm in prune step. All items in the database are reordered in support ascending order at every recursion step of the algorithm [18]. Eclat counts the supports of all itemsets more efficiently than Apriori.

Eclat is better than the Partition algorithm for using the memory. Because of the itemsets reordering process, Apriori and Eclat algorithms are used the prefix tree structure. The main differences between Apriori and Eclat are traversing to this prefix tree and determining the support of an item set.

Apriori traverses the prefix tree in breadth first order; it first finds the frequent 1-itemsets, then frequent 2 itemsets and so on [19]. Supports of the itemset are determined in two steps. Firstly, checking all transaction which contains the item sets or traversing the prefix tree for each item set. Then, incrementing the corresponding item set counters. Eclat traverses the prefix tree in depth first order, it extends an item set prefix until it reaches the limit between frequent and infrequent itemsets and then go back to work on the next prefix. Supports of the item set are determined by constructing the list of identifiers of transactions that contain the itemset. Lists of transaction identifiers of two item sets are intersecting which differ only by one item and together form the item set currently processed.

Eclat algorithm uses support based measure to find maximal frequent IS by using I-projected databases technique but this algorithm generates large number of candidate set to derive frequent item set at each iteration of the algorithm. This technique takes requires less memory compare to FP-growth to find FIS.

3.3.3 Sampling approach

To find FIS sampling approach is proposed by Toivonen [20]. In this method single sample of the database is taken from which it derives from a candidate set for full database search. The basic idea of sampling approach is to pick a random sample S of a given data D , and then search for frequent itemsets in S instead of D . The sample size of S is such that the search for frequent itemsets in S can be done in main memory, and so only

one scan of the transactions in S is required overall. Because the algorithm searches for frequent itemsets in S rather than in D , it is possible that it will miss some of the global itemsets. To lesson this possibilities, the technique is to take a lower threshold than minimum support to find the frequent itemsets local to S (denoted L^S). The rest of the database is than used to compute the actual frequencies of each itemset in L^S . if L^S actually contains all of the global frequent itemset in D , then only one scan of D is required.

The sampling approach is especially beneficial when efficiency is of utmost importance, such as in computationally intensive applications that must be run on a very basis. The penalty in sampling is that candidate set derived is necessarily a superset of the actual set of frequent itemsets and may contain many false positives.

3.3.4 MaxMiner

MaxMiner is another algorithm for finding the maximal elements [21]. It uses “search through systematic set Enumeration” mechanism and efficient pruning techniques to quickly narrow the search [22]. MaxMiner employs a breadth-first traversal of the search space; it reduces database scanning by employing a look ahead pruning strategy, i.e., if a node with all its extensions can determine to be frequent, there is no need to further process that node. It also employs item reordering heuristic to increase the effectiveness of superset-frequency pruning.

Since MaxMiner uses the original horizontal database format, it can perform the same number of passes over a database as Apriori does.

3.3.5 FP-growth

The next best known depth first algorithm is the FP-growth algorithm. FP-growth algorithm uses support-based measure to find maximal frequent item set by using I-projected databases [23]. Both approaches ECLAT, M.J. Zaki et al (1997) and FP-growth are almost similar in the case that they both generate I-projected databases. But it uses different data structure [23].

FP-growth algorithm uses FP tree structure to find FIS. FP-growth tree is memory resident and requires additional storage in every node of the FP-tree. (Because of excessive pointers storage in every node)

FP- Growth method only needs two database scans when mining all frequent itemsets. In the first scan, all frequent items are found. Next scan, constructs the first FP – tree which contains all frequency information of the given dataset.

Fp – tree use compact data structure based on the following properties [24].

- Frequent pattern generation mining performs one scan of Database to determine the set of frequent items.
- Method needs to store each item in a compact structure, thus, more than two database scan unnecessary.
- Each frequent item located in the FP – tree and each node hold items and count of the frequent item.
- Each item has to be sorted in their frequency descending order. So, tree construction operation performs easily.

3.3.6 MEDIC

Goethals presented MEDIC (Memory Efficient Discovery of Itemsets) Algorithm which generates all Itemsets containing item I as soon as there can be no transaction anymore that contain I [25]. One transaction processes at a time in lexicographic order. After generating all these itemsets, the cover of I can be removed from main memory. After that the transaction identifier of the current transaction is added to the cover of all items occurring in that transaction.

Medic is also based on support count measure and utilizes the ECLAT algorithm for mining the frequent item sets. Medic uses much less memory than Eclat because the database is never entirely loaded into main memory. Indeed, while Eclat initially stores all covers of all items in main memory, Medic only stores the covers of all items up to the currently read transaction and removes them as soon as the item can no longer occur in any forthcoming transaction. By initially reordering all items in ascending order of sup-

port, we make sure that this removal of covers happens soon, but we also make sure that the covers of very frequent items only get filled while most other covers have already been deleted. Note that the algorithm in general also performs much better, when sorting the database is not taken into account, since the intersections it needs to perform to count the support of the itemsets are applied to shorter tid's lists (or diffsets).

3.4. Problem Statement

Association rule mining comprises of two steps i.e. finding frequent itemsets (FIS) and generating association rules, based on the frequent itemsets. Finding of FIS is computationally difficult and I/O intensive. Researchers have suggested number of different techniques to find FIS. All such techniques are based on support based measure. While clustering we arrange data points so that the points nearest to each other are placed in one cluster. This can be done either by similarity or dissimilarity measures. Similar data items will be nearest to each other and dissimilar will be at distance far apart. The association rule mining task first introduced in [11] can be stated as follows:

Let $I = \{i_1, i_2, i_3, \dots, i_m\}$ be the set of items and Y be the set of transactions IDs and $i_{1 \leq j \leq m} \in I$ is $i_{1 \leq j \leq m} \subseteq Y$. Let $D = \{t_1, t_2, \dots, t_m\}$ be the transaction database where every transaction in D has unique transaction identifier in Y . Let X is item set if $X \subseteq I$ and X is frequent item-set we use the following similarity measure to find FIS;

$$\text{Sim}(x) = \frac{|\bigcap_{1 \leq j \leq m} i_j|}{|\bigcup_{1 \leq j \leq m} i_j|} \geq \phi$$

Where ϕ is user specified similarity threshold value between 0 and 1.

CHAPTER 4

Methodology

4 METHODOLOGY

In this chapter we discuss design and structure of the software. In order to implement SB-Miner we have taken into consideration number of things given in the following.

- Data structure used for SB-Miner
- Node structure used for SB-Miner
- Database layout used for SB-Miner
- Software architecture

4.1 Data structure used for SB-Miner

In order to find FIS we have chosen set enumeration tree to arrange frequent itemsets tree at different levels as shown in Figure 4.1. For the implementation purpose we have adopted the alternative representation of the set enumeration tree as shown in Figure 4.2. Every subtree as shown in the Figure 4.2 is representing an equivalence class of its root node.

Definition [28]

If \sim denotes an equivalence relation on a set A and $a \in A$ then equivalence class of "a" is the set of all elements $x \in A$ with $x \sim a$.

Every child node in the tree is the superset of its parent node and its tag starts with the class ID. Class ID i.e. tag field of the node is paired with rest of the members of the same equivalence class in lexicographic order.

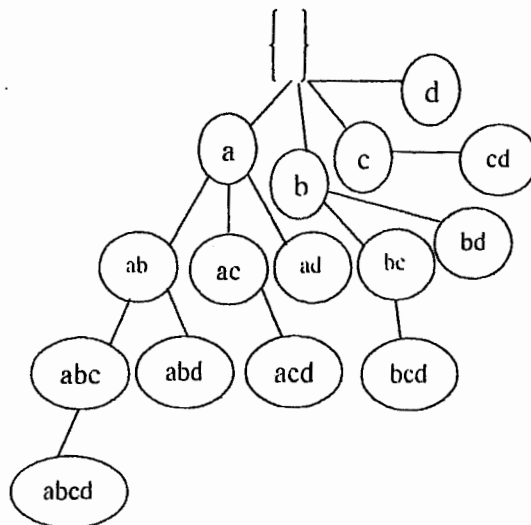


Figure 4.1 SE-Tree

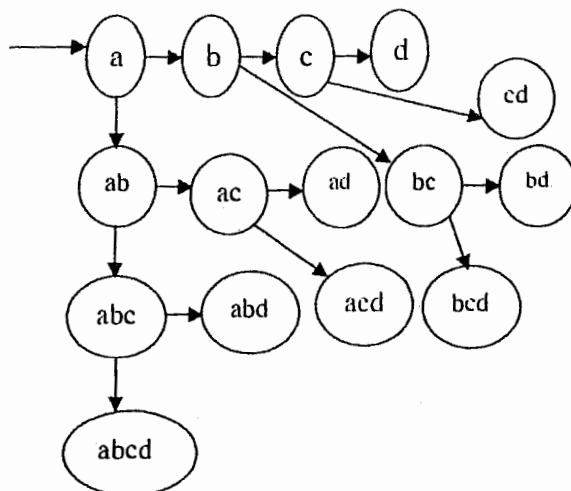


Figure 4.2 Alternative representation of SE-Tree

4.2 Node structure used for SB-Miner

Each node in SE-tree represents a frequent itemset. Each node has tag field, which shows the name of the node and also indicates frequent itemset. Node has count field indicating the total no of transactions containing this frequent itemset. There are two node pointers down and right node pointer pointing to the node that is linked to current node in the downward position and to the right position respectively. There is an info field in every node containing the transaction IDs list, which actually represents the transaction IDs of transaction containing the particular FIS represented by the tag of the node.

4.3 Database layout used for SB-Miner

Numerous data-mining algorithms have been proposed so far to find out FIS and every algorithm transforms the given database into a particular format or layout before finding FIS. Among various layouts of the database horizontal and vertical layout are very much common layouts as shown in the Figure 4.3. Horizontal layout consists of list of transactions [29]. Each transaction has an identifier followed by list of items. The vertical layout consists of list of items [30]. Each item has a transaction IDs list- the list of all the transactions containing the item. Our approach makes use of the vertical layout because we use transaction IDs lists intersection and union operation to compute the FIS. Vertical

layout of the database has a number of advantages like multiple scans of the database can be avoided. Also relevant transactions can be clustered together.

DATABASE		HORIZONTAL ITEM SET				VERTICAL TID SET				
TID	ITEMS	1	A	B	E	A	B	C	D	E
1	A, B, E	1	A	B	E	1	1	3	2	1
2	B, D	2	B	D		4	2	5	4	8
3	E, C	3	B	C		5	3	6		
4	A, B, D	4	A	B	D	7	4	7		
5	A, C	5	A	C		8	6	8		
6	B, C	6	B	C		9	8	9		
7	A, C	7	A	C			9			
8	A, B, C, E	8	A	B	C	E				
9	A, B, C	9	A	B	C					

Figure 4.3 Database layouts

4.4 Software architecture

To find FIS with SB-Miner we have created two modules given in the following.

- Preprocessing module
- FIS finding module using Jacquard similarity (SB-Miner)

4.4.1 Preprocessing module

Preprocessing step is first step in finding FIS. The preprocessing is not directly part of SB-Miner but we need this module to prepare the data set to be mined with SB-Miner. In this phase we have transformed the dataset to be mined into a format which can be input to the SB-Miner. We have converted every transaction into binary format as shown in the table 4.1.

Table 4.1 Binary Database format

TID	a	b	c	d	e	f
1	1	1	0	0	0	0
2	1	0	1	1	1	0
3	0	1	1	1	0	1
4	1	1	1	1	0	0
5	1	1	1	0	0	1

Where each row corresponds to a transaction and each column corresponds to an item. An item can be treated as binary variable whose value is one if the item is present in a transaction and zero otherwise. Because the presence of an item in a transaction is often considered more important than its absence, an item is an asymmetric binary variable.

4.4.2 FIS finding module using Jacquard similarity (SB-Miner)

To implement the FIS module using jacquard similarity measure we provide generalized steps of the module shown in figure 4.4. The algorithm as shown in Figure 4.4, takes two parameter, the data set D and minimum similarity threshold T provided by the user. To perform the step 1, **Readfilemakelist** is a module designed to form the first level of the set enumeration tree i.e. nodes of the first level of the tree are created. In this module first a node is created and then the column of data set is scanned according to the pointer named *currentP* with the help of a function named **fillbuffer**, which requires *recordlength* parameter, the *fillbuffer* fills the buffer.

Algorithm SB-Miner.

Input

D : database of transactions

T : user specified threshold.

Output F (Frequent itemsets).

ρ Number of FIS after iteration.

A Addresses of nodes having FIS.

1 Read data set D and make first level of SE-tree;

2 $\rho = \text{find_frequent_2_itemset}(F, T)$;

Repeat step 3 to 4 until $\rho = 0$

3 Assign the address of every 1st node corresponding to equivalence classes in last level of SE-tree to A ;

4 $\rho = \text{find_next_FIS}(A, F, T)$;

5 return F ;

Figure 4.4 SB-Miner's Pseudo code

This buffer is then copied by a function named **copycache2node** in to newly created node info field. This function takes the pointer to newly created node and size of the buffer and node tag to be copied into that node. The created node is then linked with the previous node and in this way the first level of set enumeration tree is created.

In step 2, *find_Frequent_2_itemset* function constructs 2nd level of the itemset tree by generating all frequent 2-itemset. Frequent 2-itemsets are formed by finding similarity of every 2 itemsets present in the first level of SE-tree. Only those items set whose similarity is greater or equal to the user supplied minimum similarity threshold are declared frequent and are linked in 2nd level of the SE-tree in lexicographic order. The FIS that are generated in the previous attempt of finding FIS can be the member of any equivalence class, and linked in the tree under its equivalence class in proper position.

To accomplish step 3 we created *Select_Previous_FIS* function, selects the FIS that are generated so far in the last level of the itemset tree under any equivalence class and stores the starting pointers of the link list corresponding to equivalence classes in last level of SE-tree in A. So far we have reached the 2nd level of the itemset tree, this level contains all the frequent itemset that are of size 2.

In order to determine the FIS of size 3 or greater we have to find the similarity among any 2 itemsets that are generated in the previous level of the itemset tree.

For this purpose we have written a function named *find_Next_FIS*, which makes use of the Inclusion Exclusion (IE) principle to generate the FIS having size greater than 2. It is best described with the help of example given in the following.

Example:-

Suppose we have 3 items A, B, C with transaction IDs

$$A = \{1, 3, 5, 7\} \quad B = \{1, 5\},$$

$$C = \{1, 7\}$$

$$A \cap B = \{1, 5\}, |A \cap B| = 2$$

$$A \cap C = \{1, 7\}, |A \cap C| = 2$$

$$B \cap C = \{1\}, |B \cap C| = 1$$

$$\text{And } A \cap B \cap C = (A \cap B) \cap (B \cap C) = \{1\}$$

$$|A \cap B \cap C| = 1$$

But

$$|A \cup B \cup C| = ?$$

In order to compute similarity of itemset A, B, C we need

$$\text{sim}_{ABC} = |A \cap B \cap C| / |A \cup B \cup C|$$

And to find $|A \cup B \cup C|$ we need IE principle

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C| = 4$$

Definition [28]

Given a finite number of finite sets, A_1, A_2, \dots, A_n , the number of the elements in union $A_1 \cup A_2 \cup \dots \cup A_n$ is where the first sum is over all i , the second sum is over all pairs i, j with $i < j$, the third sum is over all triples i, j, k with $i < j < k$, and so forth.

$$|A_1 \cup A_2 \cup \dots \cup A_n| = \sum_i |A_i| - \sum_{i < j} |A_i \cap A_j| + \sum_{i < j < k} |A_i \cap A_j \cap A_k| - \dots + (-1)^{n+1} |A_1 \cap A_2 \cap \dots \cap A_n|$$

To find the FIS of size greater than 2 *find_next_FIS* function takes starting pointers of the link lists of FIS generated in previous iteration along the user specified similarity thresholds T and FIS construct F . *find_next_FIS* function returns the total number of FIS generated in call to this function which are then assigned to ρ . *find_next_FIS* returns 0 if no FIS are created and returns none zero positive value if FIS are created. There are two pointers *inner_node* and *outer_node*. *Outer_node* pointer is assigned the address of the node from construct A points to the start of the link list and *inner_node* pointer which is pointing to the node right of the *outer_node* in this function. There are two loops nested together outer loop is controlled by the *outer_node* pointer and terminates when the *outer_node* pointer reaches to the end of the link list. Inner loop is controlled by the *inner_node* pointer and terminates when *inner_node* pointer reaches end of the link list. Outer loop is used to advance the *outer_node* pointer while inner loop is used to advance the *inner_node* pointer. In every iteration of the inner loop both of the nodes to which *inner_node* and *outer_node* pointer are pointing, their tags are combined to generate the

third node tag. To generate frequent itemsets we need to perform both intersection and union operation on TID lists of the nodes. After taking the ratio of the total number of elements in intersection and union operation If this ratio is found greater or equal to the user supplied threshold value, it is linked in the tree otherwise it is discarded and inner_node pointer moves to the next node, If present at right position of the currents node.

CHAPTER 5

Experimental Results

5. EXPERIMENTAL RESULTS

In this chapter we will discuss the technique to generate the synthetic database, results that we obtain with SB-Miner and at the end conclusion and future work.

5.1 Introduction

In order to perform experiments with SB-Miner we needed data sets. ARtool is an application for mining association in rules in binary databases [31]. This tool has utility to generate synthetic binary databases. The databases generated by using ARtool are in a specific format to be used only with this tool. But we needed to have synthetic database in format required to be used with SB-Miner. Therefore first the binary database is converted into ASCII format by the utility available in ARtool i.e. db2asc and then this ASCII format is converted into binary database format used in SB-Miner algorithm. The algorithm for conversion of data set produced with ARtool is given figure 5.1.

Algorithm Conversion_to_Binary_dataset

Input

C Number of columns

R Number of rows

ID Input data set

N Number of records

Output

OD Output data set

Repeat steps 2 to 4 until N records are transformed.

- 1 *Read field number from the current record from input data set ID.*
- 2 *For the current record put zero's in place of all absent field in the output data set OD a head from the previous field till current-1 field read from the input data set ID for the current record.*
- 3 *Put 1 in place of current field number read from the input data set ID in output data set OD for the current record.*
- 4 *If end of record is found in the input dataset ID put zero's in output data set OD starting from previous+1 field read from input data set till end of total number of fields and put end of record in output data set OD. Advance record pointer to next record in input data set OD.*
- 5 *Return output data set OD*

Figure 5.1 Data set conversion algorithm.

5.2 Synthetic database generation and results

By using the above technique we have generated synthetic databases described in the table 5.1.

Table 5.1 Synthetic binary databases

Database	T	AT	I	P	AP
T200AT6I10P5AP4.db	200	6	10	5	4
T500AT5I10P5AP3.db	500	5	10	5	3
T400AT6I10P5AP4.db	400	6	10	5	4

Where T is number of transaction AT is average size of transaction I is the number of items P is number of patterns and AP is average size of patterns. We have coded the SB-Miner in C++ language and experimented with the above data sets on 750MHZ machine with 256 MB of RAM and windows XP operating system with user supplied similarity threshold value of 0.5 to generate FIS as shown in the Figure 5.2. We have noticed that lesser number of FIS are generated as the size of FIS is increased and vice versa.

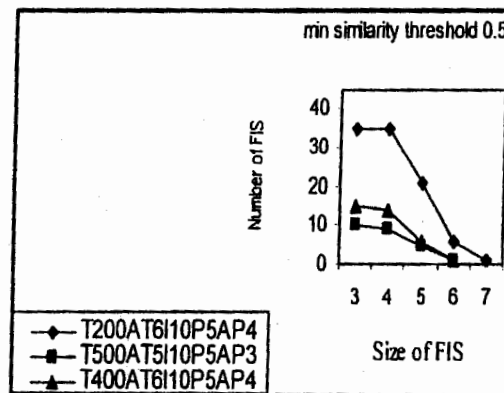


Figure 5.2 FIS generation with SB-Miner

In order to verify the correctness of our technique we have applied apriori and FP-growth on the data sets given in the table 5.1 as shown in table 5.2. Also this established that same FIS are generated with both algorithms as SB-Miner has generated as mentioned in table 5.3.

Table 5.2 Max number of FIS (0.5 similarity threshold)

Database	Algorithm	Total FIS
T200AT6I10P5AP4	Apriori	98
	FP-growth	
T500AT5I10P5AP3	Apriori	25
	FP-growth	
T400AT6I10P5AP4	Apriori	36
	FP-growth	

Table 5.3 Max number of FIS (0.5 similarity threshold)

Database	Algorithm	Total FIS
T200AT6I10P5AP4	SB-Miner	98
T500AT5I10P5AP3	SB-Miner	25
T400AT6I10P5AP4	SB-Miner	36

5.3 Conclusion

The objective of this research is to study distance based similarity measures for ARM. [32] Discusses 21 different distance based measures. Among those distance based measures, in this paper we have studied the application and implementation of Jacquard similarity measure for generating frequent item sets. Our algorithm utilizes the vertical database layout and set enumeration tree structure to arrange frequent itemsets. This also establishes that the clustering measures can also be used for association rule mining.

5.3.1 Future Work

The experimental results presented in the section 5.2 shows effectiveness of our technique. In future we intend to compare SB-Miner algorithm with the established algorithms of association rule mining for efficiency purpose.

References & Bibliography

References and Bibliography

- [1] Chen, M.-S., Han, J., and Yu, P. S. 1996. Data mining: an overview from a database perspective. *IEEE Trans. On Knowledge And Data Engineering* 8, 866-883.
- [2] Jiawei Han and Micheline Kamber. Simon Fraser University. *Data Mining: concepts and Techniques*. Simon Fraser University. August 2000. Morgan Kaufman.
- [3] Margaret H Dunham, *Data Mining Introductory and Advanced Topics*, Beijing: Tsinghua University Press, 2003: pp. 195-220.
- [4] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy. *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1996.
- [5] G.K.Gupta, *Introduction to Data Mining with Case Studies*, September 2006, Prentice Hall of India: pp. 6-7
- [6] A. El-Hamdouchi and P.Willet, Comparison of Hierarchic Agglomerative Clustering Methods for Document retrieval, the computer journal , vol.3, No 3,1989.
- [7] Gerald Kowalski, *Information Retrieval System- Theory and Implementation*, Kluwer Academic Publishers, 1997
- [8] L.Kaufman and P.J.Rousseeuw, *Finding groups in Data: an Introduction to Cluster Analysis*, John Wiley and Sons, 1990.
- [9] Kumar P., (2001), *Data Mining of Association Rules*, Computer Science and Engineering Indian Institute of Technology
- [10] Zaki M. J., (2000), *Generating Non-Redundant Association Rules*.
- [11] Agrawal R., Imielinski T., & Swami A. (1993) Mining association rules between sets of items in large databases. *SIGMOD*, 207-216.
- [12] Agrawal R. & Srikant R., (1994), Fast algorithms for mining association rules. *VLDB*, 487-499.
- [13] Savesere, A., Omiecinski, E., and Navathe, S. 1995. An efficient algorithm for mining as association rules in large databases. In *Proceedings of 20th International Conference on VLDB*.
- [14] Leung K. S. (2002), *Efficient and Effective Exploratory Mining of Constrained Frequent Sets*.

- [15] Zaki M. J., (1999), Parallel and Distributed Association Mining: A Survey. Sixth ACM SIGKDD international conference on Knowledge discovery and data mining,, 309-425.
- [16] Frans coenen, Graham GoulBourne, Paul Leng "Tree Structure for Mining Association rules "July 17 2002.
- [17] M.J. Zaki, S Parthasarathy, M.Ogihara,"New Algorithms for fast discovery of Associations Rules", Third Int'l Conf. Knowledge Discovery and Data mining, Aug .1997
- [18] Goethals B. (2003), Frequent Pattern Mining (survey), Department of Computer Science University of Helsinki.
- [19] Borgelt C., (2003), Efficient Implementations of Apriori and Éclat, Workshop of Frequent Item Set Mining Implementations (FIMI).
- [20] Toivonen, H.1996, sampling large databases for association rules. In Proc.22nd VLDB Conference, Bombay, pp. 134-145
- [21] Bayadro, R.j. 1998 Efficiently mining long patterns from databases. In pro. ACM-SIGMOD Int Conf on management of data,pp.85-93
- [22] Rymon, R.1992, search through systematic set enumeration. In Proc 3rd Int'l Conf.on principles of knowledge Representation and reasoning, pp 539-550
- [23]J. Han, J. Pei, Y. Yin, and R. Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, 2004
- [24] Goethals B. (2002) Efficient Frequent Pattern Mining, Department of Computer Science University of Helsinki.
- [25] Bart Goethals SAC', March 14-17, 2004, Nicosia, Cyprus "Memory issues in Frequent Itemset mining "
- [26] Moses Charikar , Chandra Chekuri, Tomas Feder, and Rajeev Motwani, Incremental Clustering Method for Document Retrieval, *The Computer journal*, vol.32,No. 3,1989.
- [27] Sudipto Guha, Rajeev Rastogi, and Kyaseok Shim, (1998), ROCK: A Robust Clustering Algorithm for Cateorical Attributes In proceedings of the 15th International Conference on Data Engineering, 1999.
- [28] Goodaire, G Edgar, Parmenter, M Micheal. 1998. Discrete mathematics with graph theory. Prentice-Hall

[29] Agrawal, R.; Mannila, H.; Srikant, R.; Toivonen, H.; & Verkamo, A. 1996. Fast discovery of association rules. In advances in KDD.MIT press.

[30] Holsheimer, M.; Kersten, Mannila, H.; & Toivonen, H. 1995. A perspective on database and data mining. In 1st KDD conf.

[31] L. Cristofor, College of science and mathematics UMass Boston, Dept of computer science. <http://www.cs.umb.edu/~laur/ARtool/> , 15/11/ 2007.

[32] P-N. Tan, V. Kumar, J. Srivastava. Selecting the right interestingness measure for association patterns. *Proc. KDD-2002*

Appendix A

SIMILARITY BASED MINING FOR FINDING FREQUENT ITEMSETS

M. Sikander Hayat Khiyal
m.sikandarhayat@yahoo.com

Saif UR Rehman
INTERNATIONAL ISLAMIC UNIVERSITY ISLAMABAD (IIUI) PAKISTAN
saifeyabbas@yahoo.com

Dawlat Khan
IIUI
dawlat_khan2000@yahoo.com

Abdus Salam
IIUI
abduslam@hotmail.com

Abstract - Finding association rules is a two step process. First step finds frequent itemsets (FIS), which is proved to be NP-Complete. The second step finds association rules from FIS, which is trivial. We present a novel algorithm to find FIS based on clustering measure i.e. jacquard similarity measure. Jacquard similarity measure is based on calculating the distance between itemsets. Our proposed technique makes use of prefix tree as data structure and vertical database layout to cluster related items together. The experimental results have proved that the same FIS can be generated by our technique as compared to other Apriori based algorithms. We can also show that various clustering measures can be applied for association rules mining.

Key words: Data Mining, Association Rule Mining, Similarity Measure, Frequent Itemsets.

1. Introduction

Association rule mining comprises of two steps i.e. finding frequent itemsets (FIS) and generating association rules, based on the frequent itemsets. Finding of FIS is computationally difficult and I/O intensive. Researchers have suggested number of different techniques to find FIS. All such techniques are based on *support* based measure. While clustering we arrange data points so that the points nearest to each other are placed in one cluster. This can be done either by similarity or dissimilarity measures. Similar data items will be nearest to each other and dissimilar will be at distance far apart.

The association rule mining task first introduced in [1] can be stated as follows:

Let $I = \{i_1, i_2, i_3, \dots, i_m\}$ be the set of items and Y be the set of transactions IDs and $i_{1 \leq j \leq m} \in I$ is $i_{1 \leq j \leq m} \subseteq Y$. Let $D = \{t_1, t_2, \dots, t_m\}$ be the transaction database where every transaction in D has unique transaction identifier in Y . Let X is item

set if $X \subseteq I$ and X is frequent itemset we use the following similarity measure to find FIS;

$$\text{Sim}(x) = \frac{\left| \bigcap_{1 \leq j \leq m} i_j \right|}{\left| \bigcup_{1 \leq j \leq m} i_j \right|} \geq \phi \quad (1)$$

Where ϕ is user specified similarity threshold value between 0 and 1.

The rest of paper is organized as follows. We discuss the related work in section 2; database layout used for similarity based miner (SB-Miner), itemset tree, node structure, and working of algorithm in section 3. The experimental results obtained using different data sets are discussed in Section 4. The last section relates to the conclusion.

2. Related work

Apriori is considered to be one of the primitive techniques to find FIS for association rule mining [1]. It uses support based measure and breadth first search technique to find FIS. Different Apriori versions came later on but Apriori in its original form in order to find FIS of K+1 length performs K+1 scans of the database. Max-miner algorithm uses support based measure to extract only the maximal frequent itemsets and implicitly mines all frequent itemsets [2]. Max-miner algorithm uses generic set enumeration tree (SE-tree) as data structure and performs breadth first search of the tree to limit the number of passes over the data [3]. Eclat algorithm uses support-based measure, equivalence class clustering and bottom up search method to find FIS [4]. Since this algorithm does not fully exploit the monotonicity property but generates a candidate itemset based on only two of its subsets, the number of candidate itemsets that are generated is much larger as compared to breadth first search approach such as Apriori [5]. Medic is a frequent set mining algorithm. Medic is also based on support count measure and utilizes the ECLAT algorithm for mining the frequent itemsets. FP-growth algorithm uses support measure to find maximal frequent item set by I-projected databases [6]. Both approaches ECLAT and FP-growth are almost similar in the case that they both generate I-projected database. But uses different data structure. Medic uses much less memory than Eclat because the database is never entirely loaded into main memory [5].

3. The SB-Miner

In order to find FIS we have chosen set enumeration tree to arrange frequent itemsets tree at different levels as shown in Figure 1. For the implementation purpose we have adopted the alternative representation of the set enumeration tree as shown in Figure 2. Every sub tree as shown in the Figure 2 is representing an equivalence class of its root node.

Definition [7]

If \sim denotes an equivalence relation on a set A and $a \in A$ then equivalence class of "a" is the set of all elements $x \in A$ with $x \sim a$.

Every child node in the tree is the superset of its parent node and its tag starts with the class ID. Class ID i.e. the tag field of the node is paired with rest of the members of the same equivalence class in lexicographic order.

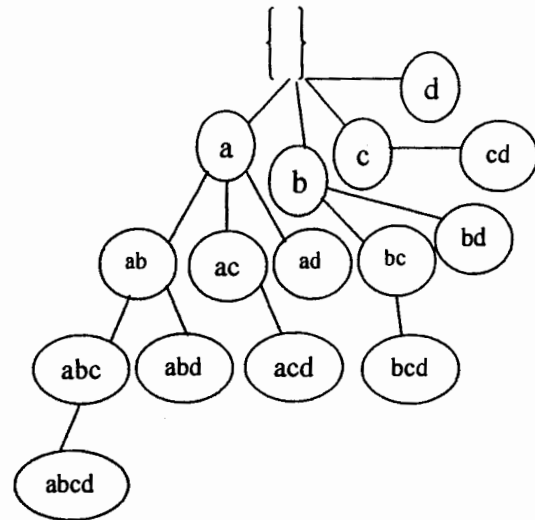


Figure 1 SE-Tree

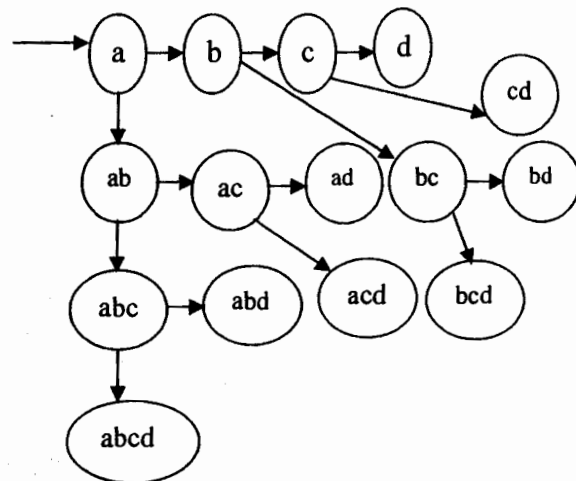


Figure 2 Alternative representation of SE-Tree

3.1. Node Structure

Each node in SE-tree represents a frequent itemset. Each node has tag field, which shows the name of the node and also indicates frequent itemset. Node has count field indicating the total no of transactions containing this frequent itemset. There are two node pointers down and right node pointer pointing to the node that is linked to current node in the downward position and to the right position respectively. There is an info field in every node containing the transaction IDs list, which actually represents the transaction IDs of transaction containing the particular FIS represented by the tag of the node.

3.2. Database layout

Numerous data-mining algorithms have been proposed so far to find out FIS and every algorithm transforms the given database into a particular format or layout before finding FIS. Among various layouts of the database horizontal and vertical layout are very much common layouts as shown in the Figure 3. Horizontal layout consists of list of transactions [8]. Each transaction has an identifier followed by list of items. The vertical layout consists of list of items [9]. Each item has a transaction IDs list- the list of all the transactions containing the item. Our approach makes use of the vertical layout because we use transaction IDs lists intersection and union operation to compute the FIS. Vertical layout of the database has a number of advantages like multiple scans of the database can be avoided. Also relevant transactions can be clustered together.

DATABASE		HORIZONTAL ITEM SET					VERTICAL TID SET				
TID	ITEMS	1	A	B	E		A	B	C	D	E
1	A, B, E	1	A	B	E		A	B	C	D	E
2	B, D	2	B	D			1	1	3	2	1
3	B, C	3	B	C			4	2	5	4	8
4	A, B, D	4	A	B	D		5	3	6		
5	A, C	5	A	C			7	4	7		
6	B, C	6	B	C			8	6	8		
7	A, C	7	A	C			9	8	9		
8	A, B, C, E	8	A	B	C	E		9			
9	A, B, C	9	A	B	C						

Figure 3 Database layouts

3.3. Working of the Algorithm

Now we provide pseudo code for the SB-Miner in the following section. The algorithm as shown in Figure 4, takes two parameter, the data set D and minimum similarity threshold T provided by the user. To perform the step 1, **Readfilemake-list** is a module designed to form the first level of the set enumeration tree i.e. nodes of the first level of the tree are created. In this module first a node is created and then the column of data set is scanned according to the pointer named **currentP** with the help of a function named **fillbuffer**, which requires **recordlength** parameter, the **fill-buffer** fills the buffer.

Algorithm SB-Miner.

```

Input
D      : database of transactions
T      : user specified threshold.
Output F (Frequent itemsets).
ρ      : Number of FIS after iteration.
A      : Addresses of nodes having FIS.

1 Read data set D and make first level of SE-tree;
2 ρ = find_frequent_2_itemset (F, T);
Repeat step 3 to 4 until ρ = 0
3 Assign the address of every 1st node
   corresponding to equivalence classes in last
   level of SE-tree to A;
4 ρ = find_next_FIS (A, F, T);
5 return F;

```

Figure 4 SB-Miner's Pseudo code

This buffer is then copied by a function named **copycache2node** in to newly created node info field. This function takes the pointer to newly created node and size of the buffer and node tag to be copied into that node. The created node is then linked with the previous node and in this way the first level of set enumeration tree is created.

In step 2, **find_Frequent_2_itemset** function constructs 2nd level of the itemset tree by generating all frequent 2-itemset. Frequent_2_itemsets are formed by finding similarity of every 2 itemsets present in the first level of SE-tree. Only those items set whose similarity is greater or equal to the user supplied minimum similarity threshold are declared frequent and are linked in 2nd level of the SE-tree in lexicographic order. The FIS that are generated in the previous attempt of finding FIS can be the member of any equivalence class, and linked in the tree under its equivalence class in proper position.

To accomplish step 3 we created **Select_Previous_FIS** function, selects the FIS that are generated so far in the last level of the itemset tree under any equivalence class and stores the starting pointers of the link list corresponding to equivalence classes in last level of SE-tree in A . So far we have reached the 2nd level of the itemset tree, this level contains all the frequent itemset that are of size 2.

In order to determine the FIS of size 3 or greater we have to find the similarity among any 2 itemsets that are generated in the previous level of the itemset tree.

For this purpose we have written a function named *find_Next_FIS*, which makes use of the Inclusion Exclusion (IE) principle to generate the FIS having size greater than 2. It is best described with the help of example given in the following.

Example:-

Suppose we have 3 items A, B, C with transaction IDs

$$A = \{1, 3, 5, 7\} \quad B = \{1, 5\}, \\ C = \{1, 7\}$$

$$A \cap B = \{1, 5\}, \quad |A \cap B| = 2$$

$$A \cap C = \{1, 7\}, \quad |A \cap C| = 2$$

$$B \cap C = \{1\}, \quad |B \cap C| = 1$$

$$\text{And } A \cap B \cap C = (A \cap B) \cap (B \cap C) = \{1\}$$

$$|A \cap B \cap C| = 1$$

But

$$|A \cup B \cup C| = ?$$

In order to compute similarity of itemset A, B, C we need

$$\text{sim}_{ABC} = \frac{|A \cap B \cap C|}{|A \cup B \cup C|}$$

And to find $|A \cup B \cup C|$ we need IE principle

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| \\ - |B \cap C| + |A \cap B \cap C| = 4$$

Definition [7]

Given a finite number of finite sets, A_1, A_2, \dots, A_n , the number of the elements in union $A_1 \cup A_2 \cup \dots \cup A_n$ is where the first sum is over all i , the second sum is over all pairs i, j with $i < j$, the third sum is over all triples i, j, k with $i < j < k$, and so forth.

$$|A_1 \cup A_2 \cup \dots \cup A_n| = \sum_i |A_i| - \sum_{i < j} |A_i \cap A_j| + \sum_{i < j < k} |A_i \cap A_j \cap A_k| - \dots + (-1)^{n+1} |A_1 \cap A_2 \cap \dots \cap A_n| \quad (2)$$

To find the FIS of size greater than 2 *find_next_FIS* function takes starting pointers of the link lists of FIS generated in previous iteration along the user specified similarity thresholds T and FIS construct F . *find_next_FIS* function returns the total number of FIS generated in call to this function which are then assigned to ρ . *find_next_FIS* returns 0 if no FIS are created and returns none zero positive value if FIS are created. There are two pointers *inner_node* and *outer_node*. *outer_node* pointer is assigned the address of the node from construct A points to the start of the link list and *inner_node* pointer

which is pointing to the node right of the *outer_node* in this function.

There are two loops nested together outer loop is controlled by the *outer_node* pointer and terminates when the *outer_node* pointer reaches to the end of the link list. Inner loop is controlled by the *inner_node* pointer and terminates when *inner_node* pointer reaches end of the link list.

Outer loop is used to advance the *outer_node* pointer while inner loop is used to advance the *inner_node* pointer. In every iteration of the inner loop both of the nodes to which *inner_node* and *outer_node* pointer are pointing, their tags are combined to generate the third node tag. To generate frequent itemsets we need to perform both intersection and union operation on tid lists of the nodes. After taking the ratio of the total number of elements in intersection and union operation If this ratio is found greater or equal to the user supplied threshold value, it is linked in the tree otherwise it is discarded and *inner_node* pointer moves to the next node, If present at right position of the current node.

4. Experimental Results

In order to perform experiments with SB-Miner we needed data sets. ARtool is an application for mining association in rules in binary databases [10]. This tool has utility to generate synthetic binary databases. The databases generated by using ARtool are in a specific format to be used only with this tool. But we needed to have synthetic database in format required to be used with SB-Miner. Therefore first the binary database is converted into ASCII format by the utility available in ARtool i.e. db2asc and then this ASCII format is converted into binary database format used in SB-Miner algorithm. By using the above technique we have generated databases described in the table 1.

Table 1. Synthetic binary databases

Database	T	AT	I	P	AP
T200AT6I10P5AP4.db	200	6	10	5	4
T500AT5I10P3AP3.db	500	5	10	3	3
T400AT6I10P5AP4.db	400	6	10	5	4

Where T is number of transaction AT is average size of transaction I is the number of items P is number of patterns and AP is average size of patterns. We have coded the SB-Miner in C++ language and experimented with the above data sets on 750MHZ machine with 256 MB of RAM and windows XP operating system with user supplied similarity threshold value of 0.5 to gen-

erate FIS as shown in the Figure 5. We have noticed that lesser number of FIS are generated as the size of FIS is increased and vice versa.

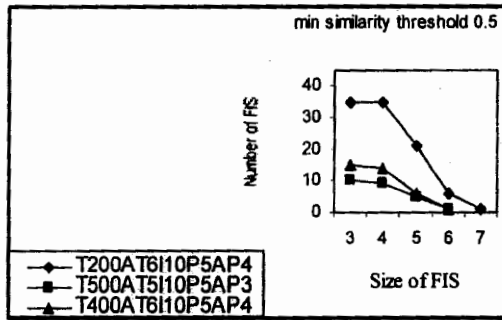


Figure 5 FIS generation with SB-Miner

In order to verify the correctness of our technique we have applied apriori and FP-growth on the data sets given in the table 1 and this established that same FIS are generated with both algorithms as SB-Miner has generated as shown in the table 2.

Table 2 Max number of FIS (0.5 similarity threshold)

Database	Algorithm	TOTAL FIS
T200AT6I10P5AP4	Apriori	98
	FP-growth	
	SB-Miner	
T500AT5I10P5AP3	Apriori	25
	FP-growth	
	SB-Miner	
T400AT6I10P5AP4	Apriori	36
	FP-growth	
	SB-Miner	

5. Conclusion

The objective of this research is to study distance based similarity measures for ARM. [11] Discusses 21 different distance based measures. Among those distance based measures, In this paper we have studied the application and implementation of Jacquard similarity measure for generating frequent item sets.

Our algorithm utilizes the vertical database layout and set enumeration tree structure to arrange frequent itemsets. This also establishes that the clustering measures can also be used for association rule mining. The experiments showed the effectiveness of the technique. In future we intend to compare SB-Miner algorithm with the established algorithms of association rule mining for efficiency purpose.

References

- [1] R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules between Sets of Items in Large Databases," ACM SIGMOD Conf. Management of Data, May 1993
- [2] Bayadro, R.j. 1998 Efficiently mining long patterns from databases. In pro. ACM-SIGMOD Int Conf on management of data, pp.85-93
- [3] Rymon, R.1992, search through systematic set enumeration. In Proc 3rd Int'l Conf.on principles of knowledge Representation and reasoning, pp 539-550
- [4] M.J. Zaki, S Parthasarathy, M.Ogihara,"New Algorithms for fast discovery of Associations Rules", Third Int'l Conf. Knowledge Discovery and Data mining, Aug .1997
- [5] Bart Goethals SAC', March 14-17, 2004, Nicosia, Cyprus "Memory issues in Frequent Itemset mining "
- [6] Han, J., J. Pei, and Y.Yin, 2004, Mining frequent patterns without candidate generation: A frequent pattern tree approach, Data mining and knowledge discovery, vol. 8, NO. 1, PP. 53-87
- [7] Goodaire, G Edgar, Parmenter, M Micheal. 1998. Discrete mathematics with graph theory. Prentice-Hall
- [8] Agrawal, R.; Mannila, H.; Srikant, R.; Toivonen, H.; & Verkamo, A. 1996. Fast discovery of association rules. In advances in KDD.MIT press.
- [9] Holsheimer, M.; Kersten, Mannila, H.; & Toivonen, H. 1995. A perspective on database and data mining. In 1st KDD conf.
- [10] <http://www.cs.umb.edu/~laur/ARtool/>
- [11] P-N. Tan, V. Kumar, J. Srivastava. Selecting the right interestingness measure for association patterns. *Proc. KDD-2002*.

[News](#)
[Travel](#)
[Finance](#)
[Entertainment](#)

[Yahoo!](#)
[My Yahoo!](#)
[Mail](#)
[Tutorials](#)
[More](#)

Welcome,
saifeyabbas
[Sign Out](#)
[All-New Mail](#)
[Help](#)

[Make Y! your home page](#)

YAHOO! MAIL
Classic

Yahoo! SearchSearch:

Web Search



Matrimo
Marriage & Dating

[Mail](#)[Addresses](#)[Calendar](#)[Notepad](#) **Mail Upgrades - Options**

[Check Mail](#)[Compose](#)

Search Mail:

Search MailSearch the Web



Hot Web Finds
Daily on THE 9

[Previous](#) | [Next](#) | [Back to Messages](#) [Call Instant Message](#)

[Delete](#)[Reply](#)[Forward](#)[Spam](#)[Move...](#)

[Folders](#)[Add - Edit] [Printable View](#) This message is not flagged. [[Flag Message](#) - [Mark as Unread](#)]

Inbox (1)

Date: Tue, 18 Sep 2007 13:48:50 +0900 (KST)

Draft

From: "ICCS" <iccs@daegu.ac.kr> [Add to Address Book](#) [Add Mobile Alert](#)

Sent

To: saifeyabbas@yahoo.com

Subject: ICCS Acceptance/Rejection Notice

Bulk (1)[Empty]

Dear Saif UR Rahman.

It is our great pleasure to inform you that your

Trash[Empty]

Paper Number: 15 30-1

Title: SIMILARITY BASED MINING FOR FINDING FREQUENT ITEMSETS

Author(s): Saif UR Rahman(INTERNATIONAL ISLAMIC UNIVERSITY ISLAMABAD), Dr M. Sikander Hayat Khiya, Abdus Salam, Dawlat Khan

Search Shortcuts

My Photos

My Attachments



Watch Christina perform live!



Emmy Awards on Yahoo! TV



Saw IV Blood Drive



Get the top 100 music videos

has been accepted for presentation at the 8th ICCCS and publication in the proceedings.

Each of the papers submitted to ICCCS 2007 were reviewed by two experts. We urge you to take into account reviewers' reports attached below to improve the camera ready copy. Your paper has been allocated maximum 6 pages in the proceedings. The final camera ready copy must be submitted by the 8th of October. The MS WORD template for your camera ready paper is available at <http://icccs.daegu.ac.kr>. Each accepted paper must have at least one author registered by the 8th of October in order to be included in the conference proceedings.

The conference program including your presentation method determined will be announced soon on the conference web. And please check the web page from time to time. We will also contact you later if your submission is selected for the publication on a relevant journal (IJAP, IJISTA, or JICR).

Congratulations again, and looking forward to seeing you at the beautiful campus of Daegu University in November.

With best regards,

Prof. Byung-Kon Hwang, General Chair of ICCCS'07
Prof. Yongtae Do, Program Chair of ICCCS'07

=====
Reviewer reports

Reviewer 1

Appropriateness to Conference: 4
Originality: 4
Technical Strength: 4
Presentation: 3
Overall Evaluation: 4

Reviewer 2

Appropriateness to Conference: 5
Originality: 4
Technical Strength: 5
Presentation: 5