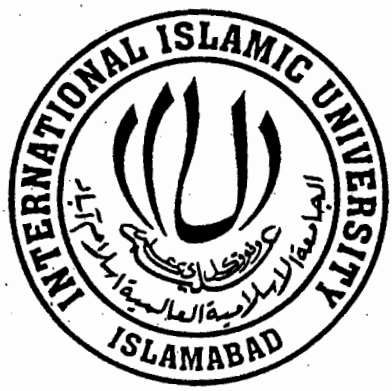
MINING FREQUENT PATTERNS WITHOUT MINIMUM SUPPORT THRESHOLD



Researcher:
Abdus Salam
Reg. No. 04-FAS/PHDCS/F03

Supervisor:
Prof. Dr. M. Sikandar Hayat Khayal

Department of Computer Science
Faculty of Basic and Applied Science
INTERNATIONAL ISLAMIC UNIVERSITY, ISLAMABAD

MINING FREQUENT PATTERNS WITHOUT MINIMUM SUPPORT THRESHOLD

Abdus Salam
Reg. No. 04-FAS/PHDCS/F03

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in computer science at the faculty of basic and applied sciences
International Islamic University,
Islamabad

Prof. Dr. M. Sikandar Hayat Khayal

April, 2010


بسم الله الرحمن الرحيم

To my family

Title of Thesis: ***Mining Frequent Patterns without Minimum Support Threshold***

Name of Student: ***Abdus Salam***

Registration No. ***04-FASA/PHDCS/F03***

Accepted by the Department of Computer Science, INTERNATIONAL ISLAMIC UNIVERSITY, ISLAMABAD, in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science.

Viva Voce Committee

Dr. Muhammad Riaz
Dean, Faculty of Basic and Applied Sciences
International Islamic University, Islamabad

Dr. Muhammad Sher
Chairman, Department of Computer Science
International Islamic University, Islamabad

Dr. Abdul Sattar (External Examiner-I)
Ex-Director General
Pakistan Computer Bureau, Islamabad

Dr. Sajjad Mohsin (External Examiner-II)
Dean, Faculty of Sciences and Information Technology
COMSATS Institute of Information Technology, Islamabad

Dr. M. Sikandar Hayat Khayal (Supervisor)
Chairperson, Department of Computer Science & SE
Fatima Jinnah Women University, Rawalpindi

Thursday, 7th April, 2011

Rudyard Kipling

I keep six honest serving-men
They taught me all I knew;
Their names are What and Why and When
And How and Where and Who.

Rudyard Kipling (1865–1936), Indian-born British writer and poet.
Just So Stories "The Elephant's Child" (1902).[1]

---

[1] *Encarta® Book of Quotations* © & (P) 1999 Microsoft Corporation. All rights reserved. Developed for Microsoft by Bloomsbury Publishing Plc.

Abstract

Finding frequent patterns is the cornerstone of many classical data mining tasks such as associations, correlations, sequences, episodes, classification, and clustering. It is also essential to a wide range of emerging applications, like Web-log mining, click-stream mining, network traffic analysis, stock market analysis, and sensor networks. Majority of existing frequent pattern mining approaches require many input parameters to be set by the users. The most widely-used parameter is the minimum support threshold to extract statistically significant patterns and to prune out the insignificant patterns. The selection of minimum support is somewhat arbitrary and there is no mechanism to ensure that this may not inadvertently remove many of the interesting patterns. Calculating support counts for the candidate itemsets consumes most of the execution time of exiting techniques. A frequent pattern mining method without minimum support threshold avoids costly candidate-generation-and-test techniques completely to give major gains in terms of performance and efficiency. This study presents a novel method to discover maximal frequent itemsets using a single database pass. Initially all 2-itemsets are generated and then *association ratio* among them is computed. Then an *association ratio graph* is constructed to facilitate the maximal frequent itemsets generation. Efficient algorithms are described for using this compact graph data structures to discover *top-most* and *top-k* maximal frequent itemsets without user specified minimum support threshold. This method employs the breadth-first search approach to construct the *all-path-source-to-destination tree* and then finds the top few maximal frequent itemsets by traversing all source to destination paths. Results are presented demonstrating the performance advantages to be gained from the use of this approach.

The frequent pattern mining framework can also be applied to solve other interesting data mining applications. The task of semantic image retrieval can be turned into a frequent pattern problem by representing image data as association ratio graph. Employing this graph-based structure, an effective semantic image retrieval architecture system is proposed for mining multimedia data efficiently.

Acknowledgements

I, dually express my deep gratitude to my supervisor Dr. M. Sikandar Hayat Khayal who helped me in the completion of my research work. I do thank him for his continuous encouragement and support for the transfer of knowledge and experience. As an advisor, he taught me practices and skills that I will use in my academic career.

I am thankful to Mr. Muhammad Sabur Seithi, President, City University of Science and Information Technology, Peshawar; without his encouragement and support I would never have reached the present state of achievement.

I would also like to thank all my teachers, colleagues, and friends at the department of computer science, International Islamic University, Islamabad for being helpful over all these years of my studies. In particular I am grateful to Dr. Muhammad Sher, head of department, for his positive comments, valuable discussions, and excellent cooperation.

Particular acknowledgements go to Dr. M. Afzal Khan, Muazzam Ali Khattak and Saif-ur-rehman for their all time assistance at critical stages for improving this work.

Let me finally acknowledge the duaas, support, encouragement, push and who constantly gave me enthusiasm that is from my mother, wife, and children. I hope I will make them proud of my achievements, as I am proud of them.

(Abdus Salam)

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AR	Association Ratio
ARCS	Association Rules Clustering System
AR-graph	Association Ratio Graph
ASD-Tree	All-path Source-to-Destination Tree
CBIR	Content-based Image Retrieval
FCI	Frequent Closed Itemset
FI	Frequent Itemset
FP-Tree	Frequent Pattern Tree
GUHA	General Unary Hypothesis Automaton
HCI	Human Computer Interaction
KDD	Knowledge Discovery in Databases
MFI	Maximal Frequent Itemset
MIS	Management Information System
MLFI	Maximal Length Frequent Itemset
MTK	Memory-constraint Top-k Frequent Pattern Mining
SAR-Graph	Semantic Association Ratio Graph
SIRS	Semantic Image Retrieval System
SQL	Structured Query Language
TKMFI	Top-K Maximal Frequent Itemset
TMMFI	Top-Most maximal Frequent Itemset
VFD	Visual Feature Database

CHAPTER 1

INTRODUCTION

1.1 The Context

"*Computers have promised us a fountain of wisdom but delivered a flood of data.*"[2] Modern societies are producing massive amount of business, economic, statistical, medical, and sports data at tremendous pace and in enormous volume. Worldwide introduction of automated data capturing devices such as barcode readers, call centers, sensors, cameras, scanners and other facilities have led to the massive increase of data stored in the databases and the data warehouses. The contemporary databases are not limited to the alphanumeric data type only rather new and diverse data types for instance image, text, sound, video, and graphic are continuously being loaded into the data repositories. However, the advancement in data collection methods has not witnessed the improved facilities in data processing and utilization. These data contain valuable hidden knowledge as rules, trends, sequences, outliers, correlations, and episodes. Data are gathered by measuring phenomena or collecting facts about objects, events, and concepts, whereas knowledge refers to data patterns and trends that are useful for decision making. The vast amount of data gathered in databases and data warehouses makes user driven

[2] Piatetsky-Shapiro, 1991

processing very time consuming and we are required to invest efforts for the self-reliant extraction of useful hidden knowledge from the data. The knowledge discovered from the very large volumes of data helps executives and managers to make knowledge-driven decisions.

1.1.1 Data Mining

Fayyad, Piatetsky-Shapiro, Smyth, and Uthurusamy (1996) define the term data mining as "non-trivial extraction of implicit, previously unknown, and potentially useful knowledge from large amounts of data". Data mining is also referred to as automated extraction of knowledge from very large databases in a reasonable time. Data mining has been established as a research area of important technological and social importance since introduction of the knowledge discovery about two decades ago. Traditional database querying model (Structured Query Language) is not appropriate to discover hidden knowledge from data. Access to a database is governed by posing queries using SQL. The query result is usually a subset of database satisfying the query conditions. On the other hand, data mining queries might not be declared explicitly and the underlying data format is also different from that of operational databases and may involve multiple passes over data. The output of the data mining query is the knowledge (rule) and not a subset of the database. Thus, data mining is completely orthogonal to the traditional database access. Data mining incorporates methods from multiple disciplines to discover hidden and interesting knowledge from the data. Data mining emerged in the late 1980s, and brings together database technology, pattern recognition, machine learning, data visualization, information retrieval, statistics, algorithms and image & signal processing.

The data mining as a rendezvous of multiple computer science disciplines is shown in Figure 1.

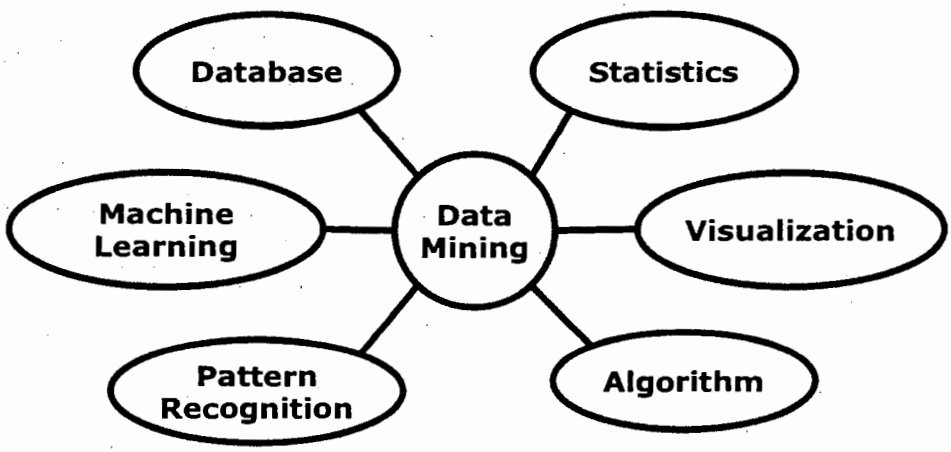


Figure 1: Data mining as rendezvous of different disciplines of computer science

Traditionally, a variety of names has been given to the concept of extracting useful patterns (the terms *itemsets* and *patterns* are used interchangeably in this text), including data mining, knowledge discovery in databases (KDD), information discovery, knowledge extraction, data archaeology, and data pattern processing. Now, the most widely used and accepted term by the data analysts, machine learning, and the management information systems (MIS) communities is the data mining. It has also become popular in the database field.

At the first KDD workshop in 1989 Piatetsky-Shapiro (1991) coined the term *knowledge discovery in databases* (KDD) to emphasize that knowledge is the end product. During the early period of data mining research data mining methods were usually applied to relational databases and data warehouses. Last decade has seen tremendous increase in structured and unstructured complex data forms such as XML, image, sound, and video. This is due to the rapid growth of data collection devices, advanced database

management systems and communication technologies. This is increasing the gap between the data and the knowledge rapidly. Therefore, the task of mining complex types of data, such as text, multimedia, spatial, bioinformatics, time-series and sequence, telecom and World-Wide Web data has increasingly become important.

1.1.2 Frequent Pattern Mining

Generally there are many types of interesting patterns that can be mined from the data. Frequent patterns are itemsets, subsequences, substructures, phrase-sets, and term-sets that appear in real-world databases with frequency no less than a user specified *minimum support* (called *min-sup* in this text) threshold. Frequent pattern mining, with over a decade of extensive research, has been established as a significant data mining task. Mining such frequent patterns is a cornerstone of various data mining tasks and applications such as association (Agrawal, Imielinski, and Swami, 1993), correlations (Brin, Motwani, Ullman, and Tsur, 1997), sequences (Agrawal and Srikant, 1995), sequence classification (Exarchos, Tsipouras , Papaloukas, and Fotiadis, 2009), episodes (Mannila, Toivonen, and Verkamo, 1997), negative rules (Wu, Zhang, and Zhang, 2004), classification (Li, Han, and Pei, 2001), and clustering (Wang, Yang, Wang, and Yu, 2002). Agrawal et al. (1993) introduced the concept of association rules along with that of frequent patterns for the market basket analysis.

Frequent pattern mining has also become a fundamental task to a wide range of emerging applications, such as Web log and click-stream mining, network traffic analysis, fraud detection, e-business and stock market analysis, and sensor networks. Association rules are applicable in many business decision-making processes, such as catalog design, target-marketing, and customer shopping behavior analysis. Another important problem

in data mining is mining frequently occurring ordered events called sequential pattern mining (Agrawal and Srikant 1995). Mannila et al., (1997) propose mining frequent episodes in a sequence of events. The application of frequent episodes involves customer shopping sequences, clickstreams, and DNA sequences. Frequent patterns have also shown to be useful for classification (Li, Dong, and Ramamohanrarao, 2000; Li and Wang, 2006; Dong and Li, 1999; Li, Han and Pei, 2001; Yin and Han, 2003; Wang and Karypis, 2005), where strong associations between frequent patterns are helpful in discovering class labels. Lent, Swami and Widom (1997) propose a system called Association Rule Clustering System (ARCS) for mining binned intervals and then clustering mined quantitative association rules for concise representation. Moreover, frequent pattern mining is essential in many other data mining tasks such as mining spatial association rules (Koperski and Han, 1995), mining of max-patterns (Bayardo, 1998), causality (Silverstein, Brin, Motwani, and Ullman, 1998), multi-dimensional patterns (Lent, Swami and Widom, 1997; Kamber, Han and Chiang, 1997), partial periodicity (Han, Dong and Yin, 1999), emerging patterns (Dong and Li, 1999), and calendric market basket analysis (Ramaswamy, Mahajan, Sillberschatz, 1998). Thus frequent pattern mining has become a focused theme of research over the last decade.

1.2 Motivation

Most of the classic frequent pattern mining approaches attempt to discover all distinct valid frequent patterns with frequency no less than a user specified *min-sup* threshold. Apart from *min-sup* users are also required to tune many other input parameters before running a mining algorithm. Keogh, Lonardi and Ratanamahatana (2004) observe that

there may arise two problems while working with parameter-laden algorithms, *i*) setting inefficient parameter values may cause an algorithm not to discover interesting patterns, and *ii*) an algorithm may report spurious patterns that do not really exist. Particularly, this is possible when a data analyst misunderstands the function of an input parameter in mining process or may not select efficient parameter values causing an algorithm to fail in discovering highly correlated itemsets. These are the motivating factors for continuing research to find efficient parameter-free algorithms for mining frequent patterns. Thus, a true data mining algorithm should require as minimum number of parameters as possible, ideally none. A parameter-free algorithm will be free of an individual's prejudices and presumptions on the problem, and allows the data to speak for itself.

However, majority of the frequent pattern mining algorithms suffers from some inherent limitations:

i. Majority of the existing frequent pattern mining algorithms is based on the Apriori algorithm (Agrawal and Srikant, 1994). This class of algorithm attempts to improve the efficiency by reducing the number of passes over the database; reducing the size of the database by using compact data structures and pruning the candidates by various techniques. However, the Apriori based algorithms still suffer from: *i*) the complex process which generates candidate itemsets that use most of the time, space, and memory, and *ii*) multiple database scans.

ii. FP-Tree (Han and Pei, 2000) is an important milestone in the development of frequent pattern mining algorithms to solve the problems posed by the Apriori class of algorithms. This technique generates frequent itemsets using only two database scans and without generating candidate itemsets. However, FP-Tree is

not suitable to be used in interactive mining circumstances. During this process, a data analyst may change the *min-sup* threshold that may lead to repetition of the whole mining process. The inability of FP-Tree algorithm for incremental mining is it's another limitation. With the passage of time databases keep growing by inserting new datasets into the database and being updated frequently that also leads to a repetition of the whole mining process.

iii. Pattern growth methods are effective in mining the dense databases as the hyper-structures encapsulate data items concisely. However, the size of hyper-structures grows exponentially with the increase in the number of database items, due to the less sharing of common prefixes.

iv. *What is the appropriate minimum support?* To determine the *min-sup* is usually left unresolved to users. Setting *min-sup* value is very subtle and quite imperative. Setting a *min-sup* to a small value may cause an algorithm to produce an extremely large number of frequent itemsets at the cost of execution efficiency. Conversely, fixing a large min-sup significantly reduces the number of discovered itemsets, but may result in losing useful information for decision making. In order to obtain true patterns, users are required to tune the min-sup over a wide range of values i.e., between 0.01% and 99.99%. This is very time-consuming and indeed is a serious problem for the applicability of mining frequent itemsets (Chuang et al., 2008).

v. Han, Cheng, Xin, and Yan (2007) consider that the "bottleneck of frequent pattern mining is not on whether we can derive the *complete* set of frequent patterns under certain constraints efficiently but on whether we can derive a *compact but*

high quality set of patterns that are most useful in applications. The set of frequent patterns derived by most of the current pattern mining methods is too huge for effective usage. Since there are usually a large number of distinct single items in a typical transaction database, and their combinations may form a very huge number of itemsets, it is challenging to develop scalable methods for mining frequent itemsets in a large dense transaction database."

vi. The number of itemsets tested during the candidate generation process is very large. The intrinsic cost of candidate (*1*, *2*, …, *n*)-itemsets generation is $\sum_{i=1}^{n}\binom{n}{i}=2^n-1$. The size of candidate sets is effectively reduced by Apriori heuristics thus achieving good performance. However, Apriori class of algorithms may still be not reasonably efficient in case of long patterns and low min-sup values.

vii. Majority of candidate generation and pattern growth approaches require multiple database scans. It is also understood that multiple database passes and checking large sets of candidates is a very time consuming process. Ideally a data mining algorithm should scan a database only once.

viii. In frequent pattern mining there is another misunderstanding that scanning the database is a very time consuming operation. However, according to Goethals (2005) this is not the major cost incurred in this class of algorithms. Instead, updating the support counts of all candidate itemsets is the more time consumable operation than reading that transaction from a database. Thus, the size of

database does affect the time needed for support counting, but it is by no means the major factor.

Since it is established that discovering frequent patterns is an essential data mining task, developing efficient methods has been recognized as an important research problem. In the previous studies there remain many unanswered questions that need to be answered.

- The core algorithm for most of candidate generation approach is Apriori, that is based upon Apriori heuristic, i.e., if any length *k* itemset is not frequent in the database, its length (*k* + *1*) superset can never be frequent. This helps in reducing the number of candidates significantly. On the other hand another class of algorithm called pattern growth technique does not require candidate itemsets generation and uses complex *hyperstructure* (FP-Tree) for mining frequent itemsets. However, during exploration both approaches use the most common constraint, i.e., *min-sup,* to reduce the itemsets. Although, *min-sup* reduces itemset identification to the finding of only those itemsets that exceed a specified presence within the database, but tuning an appropriate *min-sup* for different datasets is not an easy task. *Is there any way to improve the efficiency of frequent pattern mining by eliminating the user specified minimum support threshold?*
- Have the researchers exhausted their search for parameter-free mining techniques so that one can readily discover a compact set of interesting patterns with satisfactory performance?
- Can we discover novel frequent pattern mining method independent of candidate generation (Apriori based) and pattern-growth (FP-growth) methodologies? Can we improve upon data structures used for frequent pattern methods?

- Frequent pattern mining has been applied to a number of interesting applications. Can this method be extended to resolve other interesting data mining problems such as semantic image retrieval and image annotation?
- Can parameter-free mining frequent pattern approach help in bridging the semantic gap in the field of semantic image retrieval? Can we develop system architecture for the content based image retrieval based on frequent pattern mining paradigm?

This work goes part of the way in addressing the concerns and issues raised above.

1.3 Contributions

In this dissertation we study the problem of frequent pattern mining without user provided *min-sup* threshold. Apart from its application to market basket data its application to new and interesting problems such as semantic image retrieval is studied as well. We can summarize the contribution of this work as follows:

- While previous works improve the efficiency by mostly concentrating on reducing the database passes and using complex data structures, an attempt is made to improve the efficiency by eliminating the user provided *min-sup* threshold from the mining process.
- While previous works have employed directed graph structures, in this work an undirected weighted graph structure is used that greatly reduces the storage requirements and the processing time.

- The performance of most methods suffers due to multiple database scans. In this work it is shown that only a single database pass can produce the results and achieve considerable improvement in terms of speed and efficiency.
- An association measure based on *odds ratio* is proposed to convert the co-occurrence frequency counts of database items into proximity measure validating the most associated items and nullifying the least associated itemsets.
- To show the applicability of this technique to other interesting problems, we have also dealt with a semantic image retrieval system's architecture based upon frequent pattern mining framework.

1.4 Organization of the Thesis

This dissertation is organized as follows:

- In Chapter 2, we present formal specification of the problem of frequent pattern mining and semantic image retrieval.
- In Chapter 3, we review the relevant literature concerning frequent pattern mining as well as specific literature concerning semantic image retrieval.
- In Chapter 4, the development of a novel parameter-free frequent pattern mining technique is discussed. The efficiency and accuracy are established by theoretical analysis and experimental studies. According to the performance evaluation study our method shows good efficiency in mining large sparse databases.
- We also extend the application of frequent pattern mining without support threshold to bridge the semantic gap in semantic image retrieval problem in Chapter 5. The study shows that the proposed method can effectively be applied

to the problem of searching in the large image repositories according to their content.

- The characteristics of the proposed methods are summarized in the Chapter 6. Some interesting extensions and applications are also discussed.
- This work is concluded in Chapter 7 in which some future directions are also presented.

CHAPTER 2

PROBLEM DEFINITION

In this chapter basic notions pertaining to the problem of frequent pattern mining and semantic image retrieval are defined. In Section 2.1, the task of mining frequent patterns, its usefulness and problems that arise due to the reliance on *min-sup* threshold are discussed. In Section 2.2, we discuss current research issues regarding semantic image processing. Our emphasis is to map the parameter-free frequent pattern mining technique to the semantic image retrieval problem.

2.1 Frequent Pattern Mining

Agrawal et al., (1993) first proposed the problem of frequent itemset mining in the form of association rules for the market basket analysis. Association rules analyze buying behavior of customers by deriving associations among different items purchased together on a per-transaction basis by customers. It is a two-step process; i) identification of *frequent itemsets* within the dataset and ii) the derivation of inferences from the frequent itemsets. Most of the real world data used for frequent pattern mining is modeled in the form of set-valued attributes. A set-valued attribute contains multiple unordered values of numbers or characters. Set-valued attributes appear in a number of application domains such as set of items bought together by a customer in market basket, set of objects contained in an image in multimedia databases, and set of clicks in Web logs

representing the Web pages and links visited by a user. This set-valued attribute data is then employed in different data mining applications to discover frequent patterns, sequential patterns, and episodes.

Finding frequent itemsets is recognized as computationally most challenging task and has shown to be *NP-Complete* (Gunopulos et al., 2003). Thus, majority of the methods have focused upon efficient discovery of frequent itemsets. Given n distinct items within a database there are 2^n possible combinations of itemsets to explore. This is where the problem arises because the number of possible combinations is very large. Consequently, it is not practical to generate and test every possible combination of items and require some heuristic strategies to reduce the search space. Different heuristics proposed to limit the number of itemsets to a considerable range are discussed below.

2.1.1 Basic Notions

A formal specification of the frequent pattern mining problem is presented as follows: Let $I = \{ i_1, i_2, \ldots, i_m\}$ be a set of m distinct items. A database $D = \{t_1, t_2, \ldots t_k\}$ consists of a set of k transactions, where each transaction T is a subset of I or $T \subseteq I$. Each transaction has a unique identifier (*tid*) and contains a set of items. A set $X \subseteq I$ is called an *itemset* and $|X|$ is referred to as length of an itemset. Itemsets of length m are referred to as *m-itemset.* The number of transactions that contain an itemset is known as the occurrence frequency of that itemset. This is also called the *support count* of the itemset. The support of an itemset X, denoted by $supp(X)$, is the fraction of transactions containing X, or $supp(X) = \frac{1}{n}\sum_{i=1}^{n} 1(X \subseteq T_i)$, where, each T_i is a set of items, and $(X \subseteq T_i)$ is 1 when $X \subseteq T_i$ and 0 otherwise. Itemset X is designated as a frequent pattern if the $supp(X)$ is

larger than a user specified threshold called *min-sup*. This shows that the presence of itemset X in the database is statistically significant. The problem of frequent pattern mining can now be defined as discovering all itemsets present in the database having support higher or equal to a user specified *min-sup*. A formal definition of frequent itemset is as follow:

Definition 1 (frequent itemsets (**FI**)) Given the *min-sup* threshold, an itemset X is called frequent itemset where $supp(X)$ is greater than *min-sup* i.e. $F_I = \{X \mid supp(X) \geq min\text{-}sup\}$.

Frequent pattern mining poses a great challenge when majority of current techniques produce a large number pattern sets satisfying *min-sup* especially when min-sup is set low. Han et al (2007) observe that "this is because if a pattern is frequent, each of its subpatterns is frequent as well. A large pattern will contain an exponential number of smaller, frequent sub-patterns." This makes identifying useful patterns extremely difficult because the number of interesting itemsets is usually very small.

2.1.2 Reducing Search Space

The size of frequent itemsets discovered by most of the existing techniques is too huge for any effective usage. Many alternatives has been proposed for reducing such a huge result set such as mining *closed patterns, maximal patterns, approximate patterns, condensed pattern bases, representative patterns, clustered patterns,* and *discriminative frequent patterns* (Han et al, 2007). Most of the current methods use the anti-monotonicity property of itemset support, i.e., every subset of a frequent itemset must be

frequent. Concerted efforts are being made to enhance the quality of output patterns significantly by reducing the size of discovered pattern sets.

The discovery of large number of FIs is sometimes reduced to a small number of compact itemsets by mining only closed frequent itemsets. Pasquier, Bastide, Taouil, and Lakhal (1999) first proposed the mining of frequent closed itemsets (FCI, defined below in Definition 2) algorithm called A-Close to minimize the size of discovered patterns. CLOSET (Pei, Han, and Mao, 2000), CHARM (Zaki and Hsiao, 2002) and CLOSET+ (Wang, Han, and Pei, 2003) are the other important FCI algorithms. Mining FCIs provides an interesting alternative to mining FIs since it possesses the same analytical power but derives much lesser number of resultant itemsets.

Definition 2 (frequent closed itemsets (**FCI**)) "A pattern α is a *closed frequent pattern* in a data set D if α is frequent in D and there exists no proper superset β such that β has the same support as α in D" (Han et al., 2007).

In proposals for further reducing the size of discovered itemsets Bayardo (1998) developed an Apriori-based method called MAX-Miner. This approach calls for generation of maximal frequent itemset (MFI, defined below in Definition 3). MAFIA proposed by Burdick, Calimlim, and Gehrke (2001) is another important technique that generates maximal frequent patterns efficiently. Yang (2004) provides the theoretical analysis of the complexity of mining MFIs.

Definition 3 (maximal frequent itemsets (**MFI**)) "A pattern α is a *maximal frequent pattern* (or *max-pattern*) in set D if α is frequent, and there exists no superset β such that $\alpha \subset \beta$ and β is frequent in D" (Han et al., 2007).

Both FCIs and MFIs effectively reduce the set of derived patterns. The number of FCIs is significantly smaller than the number of all FIs, and the number of MFIs is smaller than the number of FCIs. To further reduce the size of derived pattern set Tianming, Sam, Hui, and Qian (2007) propose the idea of maximum length frequent itemset that is defined as below.

Definition 4 (maximum length frequent itemset (**MLFI**)) "An itemset X is a maximum length frequent itemset if $supp(X) \geq min\text{-}sup$ and for all itemset Y, if $supp(Y) \geq min\text{-}sup$ then $|X| \geq |Y|$, where $|Y|$ and $|X|$ denote the number of items contained in Y and X respectively" (Tianming et al., 2007).

It can be clearly seen that the following relationship exists: **MLFI** $\subseteq$ **MFI** $\subseteq$ **FCI** $\subseteq$ **FI**. The set MLFI is smaller than the set MFI. MFI is orders of magnitude smaller than the set FCI, which itself is orders of magnitude smaller than the set FI. The relationship is also depicted diagrammatically in Figure 2. When the length of derived pattern is very long it is often not feasible to construct the complete set of frequent itemsets. In biological applications such as finding combinatorial patterns generating the set of maximal frequent patterns is also adequate.

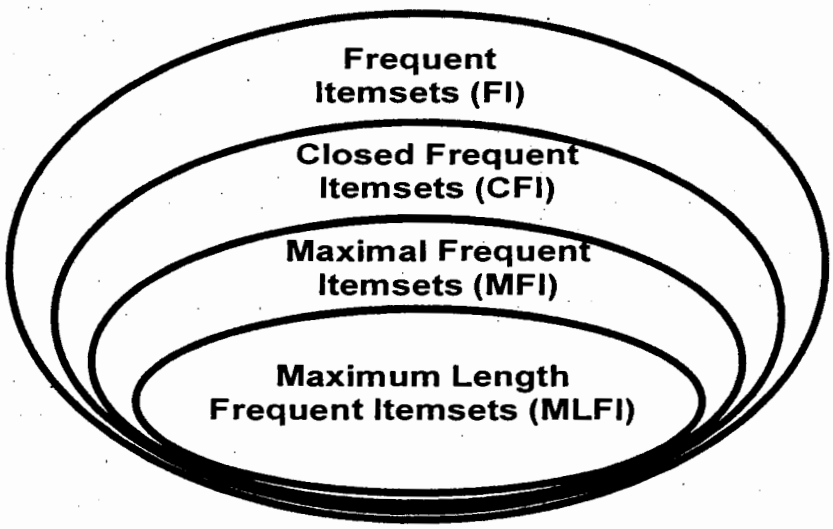


Figure 2: The hierarchy of frequent itemsets, frequent closed itemsets, maximal frequent itemsets and maximum length frequent itemsets

The use of FIs, and FCIs is restricted due to the high computational cost as well as large number of resulting itemsets in many applications. Instead, it is often required to derive a small subset of frequent itemsets, such as MFIs. MFI can efficiently be discovered even in very large databases because the number of maximal frequent itemset is usually very small. The users are mostly interested in discovering the maximal frequent itemsets and any frequent pattern mining algorithm which directly finds all MFIs is efficient and optimal. Thus the problem of finding MFIs in large databases can easily be transformed into mining *top-most* and *top-k* maximal frequent itemset. This mining process should be quick, efficient and without user provided parameters.

2.1.3 Role of Minimum Support

When surveying the literature on frequent pattern mining, it is observed that researchers while conducting empirical studies tune *min-sup* arbitrarily. There has been no criterion as how to adjust *min-sup* for different approaches to test on various standard datasets. A wide range of *min-sup* values is used to analyze the performance of various frequent

pattern mining algorithms. For example, Gouda and Zaki (2005) in their experimental analysis use the *min-sup* ranges between 70 and 20, 100 and 40, 100 and 0, 50 and 0, 0.16 and 0, 2 and 0.2, and 10 and 0.01 on the chess, pumsb, connect, pumsb*, T10I4D100K, T40I10D100K, and mushroom datasets respectively. Similarly, Burdick et al. (2001) use *min-sup* ranges between 3 and 0, 50 and 0, 0.35 and 0, 80 and 30, 0.03 and 0, 50 and 0, 0.035 and 0 and 35 and 0 on T40I10D100K, chess, BMS-POS, pumsb, BMS-WebView-1, connect4, BMS-WebView-2 and pumsb* datasets respectively to study the efficiency of their approach. It is difficult to recognize as to how *min-sup* value is adjusted for different datasets. In any case, there are clearly problems in setting parameters and reporting results on the same datasets. The impending asymmetry in tuning parameters prevents us from evaluating the contribution of many papers. Thus, the parameter-laden algorithms can have their parameters tuned to achieve excellent performance on one dataset, but may fail to produce similar results on a new but very similar dataset.

In previous works the determination of *min-sup* value is usually left to the users. Setting a suitable *min-sup* threshold even under the supervision of an experienced miner is a quite subtle task. A small *min-sup* may directly hamper the efficiency of an algorithm by generating extremely large size of frequent itemsets. Contrary to this setting, a large *min-sup* may only generate a small number of itemsets. Users are required to tune the *min-sup* over a wide range of values to discover the interesting patterns. Chuang et al. (2008) conclude that "this is very time-consuming and indeed a serious problem for the applicability of mining frequent itemsets".

Let's demonstrate the problem with the help of a concrete example based on the transaction database of Table 1. The database contains 16 items, I = {A, B, C, D, E, F, G,

H, I, J, K, L, M, N, O, P} and 25 transactions. Apriori is the fundamental and most influential algorithm for mining frequent itemsets. An illustrative example is given below to exhibit the execution of Apriori algorithm on the sample database of Table 1. The frequent itemsets are generated by setting *min-sup* value to 25%.

Table 1: An example of transaction database

TID	Items
01	A B D F P
02	B C D F
03	D E G I J
04	B C D F I
05	B C E G I
06	J O
07	A B D F J
08	H J O
09	B C D E F
10	B C E G I
11	B D I
12	B D F G I
13	A B C
14	C D E G I
15	E F
16	G
17	G H I
18	A B D F
19	B C E G I
20	D E G I
21	J O P
22	B C D F
23	E G I J K
24	K L M
25	M N O

A step-by-step generation of candidate itemsets and frequent itemsets, setting *min-sup* value to 7 is shown in Figure 3.

1. In the first iteration of the algorithm, each item is a member of the set of candidate 1-itemsets, C_1. The algorithm simply scans all of the transactions of D in order to compute the support count of each item by producing C_1.

2. The required *min-sup* count is 7 (i.e., *min-sup* = $\lceil 25 * .25 \rceil = 7$). The set of frequent 1-itemsets, L_1, can then be determined. It consists of the candidate 1-itemsets satisfying *min-sup*.
3. To discover the set of frequent 2-itemsets, L_2, the algorithm uses $L_1 \times L_1$ to generate a candidate set of 2-itemsets, C_2. C_2 consists of $\binom{|L_1|}{2}$ 2-itemsets.
4. Next, the database is scanned and the support count of each candidate itemset in C_2 is accumulated, as shown in the middle table of the second row in Figure 3.
5. The set of frequent 2-itemsets, L_2, is then determined, consisting of those candidate 2-itemsets in C_2 having *min-sup*.
6. The candidate itemset C_3 is computed as $C_3 = L_2 \times L_2$. Based on the Apriori heuristic that all subsets of a frequent itemset must also be frequent, we can determine that length three candidates cannot possibly be frequent, thereby removing them from C_3.
7. The transaction database is scanned in order to determine L_3, consisting of those candidate 3-itemsets in C_3 with support count greater than or equal to *min-sup*.

The entire process of generation of FIs is detailed in Figure 3.

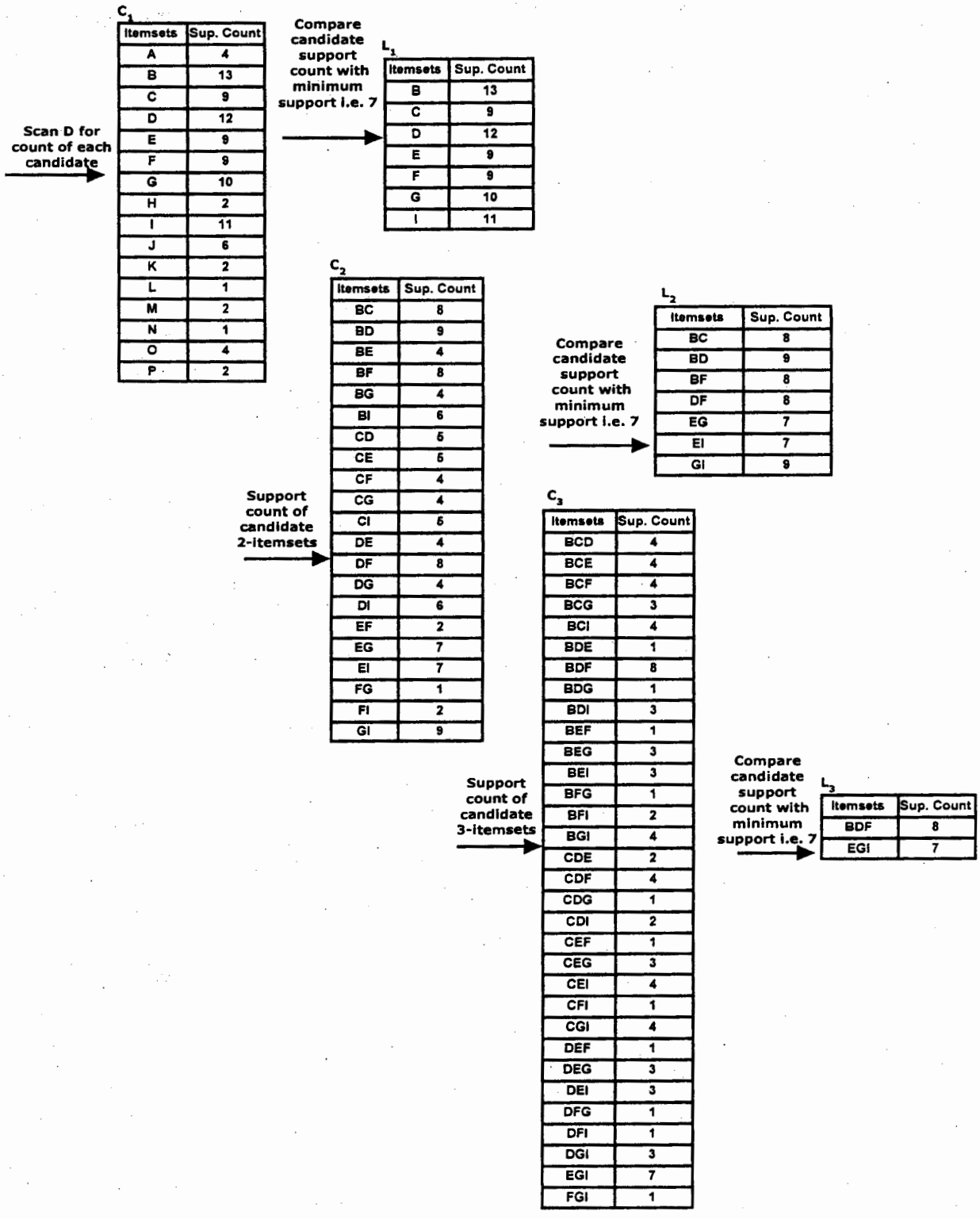


Figure 3: Generation of frequent itemsets using Apriori at 25% *min-sup*

The *min-sup* has a major influence on frequent pattern result sets. The number and size of generated frequent itemsets vary if *min-sup* threshold is changed. This is shown in the Figure 4. The results of FIs, FCIs and MFI generated at different *min-sup* values such as 15%, 25% and 35% are shown in Figure 4-a, 4-b, and 4-c respectively.

At 15% Minimum Support				
Itemsets	Support Count	FI	FCI	MFI
A	6	Yes	No	No
B	13	Yes	No	No
C	9	Yes	No	No
D	12	Yes	No	No
E	8	Yes	No	No
F	3	Yes	No	No
G	10	Yes	No	No
I	11	Yes	No	No
J	4	Yes	No	No
AC	4	Yes	No	Yes
AG	4	Yes	No	Yes
BC	5	Yes	No	No
BD	9	Yes	No	No
BE	4	Yes	Yes	No
BF	3	Yes	Yes	No
BG	4	Yes	Yes	No
BI	6	Yes	No	No
CD	4	Yes	No	No
CE	3	Yes	No	No
CF	3	Yes	Yes	No
CG	4	Yes	Yes	No
CI	5	Yes	No	No
DE	4	Yes	No	Yes
DF	3	Yes	Yes	No
DG	4	Yes	Yes	No
DI	6	Yes	No	No
EG	3	Yes	Yes	No
FI	3	Yes	Yes	No
GI	5	Yes	No	No
BCD	4	Yes	Yes	No
BCE	3	Yes	No	Yes
BDG	4	Yes	Yes	No
BFI	3	Yes	No	Yes
BGF	3	Yes	No	Yes
DEI	3	Yes	No	Yes
DGF	3	Yes	Yes	No
CEG	3	Yes	Yes	No
CFI	3	Yes	Yes	No
CGI	4	Yes	Yes	No
DGI	4	Yes	No	Yes
FGI	3	Yes	No	No
BCDF	3	Yes	No	Yes
CFGI	3	Yes	No	Yes
Total		43	15	9

(a)

At 25% Minimum Support				
Itemsets	Support Count	FI	FCI	MFI
B	13	Yes	No	No
C	9	Yes	No	No
D	12	Yes	No	No
E	8	Yes	No	No
F	3	Yes	No	No
G	10	Yes	No	No
I	11	Yes	No	No
BC	5	Yes	No	Yes
BD	9	Yes	No	No
BE	4	Yes	Yes	No
BI	6	Yes	Yes	No
BG	4	Yes	Yes	No
BF	3	Yes	Yes	No
DI	6	Yes	No	No
BDI	4	Yes	No	Yes
BCD	4	Yes	No	Yes
Total		16	4	3

(b)

At 35% Minimum Support				
Itemsets	Support Count	FI	FCI	MFI
B	13	Yes	No	No
C	9	Yes	No	Yes
D	12	Yes	No	No
E	8	Yes	No	Yes
F	3	Yes	No	Yes
G	10	Yes	No	No
I	11	Yes	No	No
BD	9	Yes	No	Yes
DI	6	Yes	No	Yes
Total		9	0	5

(c)

Figure 4: The effect of varying minimum support values on FI, FCI, MFI

.s can be seen in Figure 4 for length 1-itemset, 2-itemset, 3-itemset, and 4-itemset 10, 19, 12, and 2 number of result sets are generated respectively at 15% *min-sup* value. Similarly, at 25% *min-sup* value 7, 7, 2, and zero number of result sets are discovered for length 1-itemset, 2-itemset, 3-itemset, and 4-itemset respectively. The result set at 35% min-sup is very small as only 7 and 2 number of itemsets are produced for the length 1-itemset and 2-itemset respectively.

It is further observed that total number of FIs generated at 15%, 25% and 35% *min-sup* values are 43, 16 and 9 number of result sets respectively. The number of FCI produced at 15%, 25% and 35% *min-sup* values are 15, 4 and zero respectively. Similarly, number of MFI produced at 15%, 25% and 35% *min-sup* values are 9, 3 and 5 respectively. This shows that with the increase in *min-sup* value the number and size of result set decreases. The itemset {B, D, F} can best illustrate the inconsistency of support based approaches. At 15% support {B, D, F} is frequent but neither closed nor maximal, whereas, at 25% it becomes maximal. Further, it is not even frequent at 35% *min-sup*. It is evident in Figure 4 that $|MFI| < |FCI| < |FI|$. This shows that a user without specific domain knowledge may not set an appropriate *min-sup* to obtain true patterns. Thus a user needs to tune the *min-sup* over a wide range of values.

The basic definitions of FI, FCI, MFI and MLFI are all based on *min-sup* threshold. Mining frequent patterns in a large search space makes full use of the pruning power of the *min-sup* to cut down the search space. Thus, an attempt to eliminate *min-sup* from the mining process must be independent of pruning techniques.

2.1.4 Top-*k* Maximal Frequent Itemsets

The recent advances in data mining (Cheung and Fu, 2004, Wang, Han, Lu, and Tzvetkov 2005, Chuang, Huang, and Chen, 2008, Wang and Karypis, 2005) introduce another interesting parameter called *top-k* which refers to the *k* most frequent itemsets in the database. Cheung and Fu (2004) call for specifying the desired count (*k*) of frequent itemsets, a more easily applicable parameter, instead of specifying the *min-sup* threshold. However, the *top-k* parameter is subject to different implications in different studies, for example, Wang et al., (2005) employ *top-k* parameter to mine frequent closed itemsets, Cheung and Fu (2004) retrieve *k most frequent l*-itemsets, and Chuang et al. (2008) uses *top-k* parameter for mining frequent itemsets and closed itemsets.

The objective of majority of frequent pattern mining algorithms is to discover highly correlated itemsets in very large databases that are usually in very small number, or may be even one. Therefore, the problem of finding highly correlated patterns without user specified *min-sup* value could easily be transformed into mining *top-most* and *top-k* frequent itemsets. This mining process should be quick, efficient and without user provided parameters. We then have some necessary definitions as follows:

Definition 5 (*top-most* maximal frequent itemset (TMMFI)) An itemset X referred to as *top-most* maximal frequent itemset if for all other frequent itemsets Y the $supp(X) > supp(Y)$ and the $|X|$ and $|Y|$ is greater than or equal to three, where $|Y|$ and $|X|$ denote the number of items contained in Y and X respectively.

Definition 6 (*top-k* maximal frequent itemset (TKMFI)) Given the desired value of *k*, itemsets ($X_1, X_2, \ldots, X_k$) are *top-k* maximal frequent itemsets in *D* such that the $supp(X_1) \geq supp(X_2) \geq \ldots \geq supp(X_k)$ and the |X| is greater than or equal to three.

The reason for restricting the *top-most* and *top-k* itemsets to a minimum number is that all subsets of a MFI are also frequent. Once a large MFI is discovered then it is easy to find smaller FI. Our goal in this work is to discover the *top-most* and *top-k* maximal frequent patterns without user specified *min-sup* directly and efficiently. This can be achieved by computing association among the items using their co-occurrence frequency counts. Thus, the selection of right interestingness measure is imperative to capture the association among the items.

2.1.5 Association Measure

The core of frequent pattern mining is to discover itemsets that appear together frequently in database transactions. This task requires an appropriate symmetric measure to evaluate the correlation between co-occurring database items. In the previous works many symmetric association measures have been used widely to capture the proximity relationship among the items. For more information about different symmetric association metrics, readers are referred to Pang-Ning, Vipin and Jaideep (2004). Pang-Ning et al. (2004) observe that "although there are numerous measures available for evaluating association patterns, a significant number of them provide conflicting information about the interestingness of a pattern, and the best metric to use for a given application domain is rarely known". Pang-Ning et al., have shown that no measure is uniformly suitable in all application areas. This is because each metric is designed in

such a way as to be applicable to a specific application domain and has its own diversified characteristics. There are several well-known symmetric association measures available for computing the similarity between a pair of items. These include interest factor (I) (Brin et al., 1997), cosine (IS) (Tan and Kumar, 2000), Jaccard (ζ) (Van Rijsbergen, 1979) and Piatetsky-Shapiro (PS) (Piatetsky-Shapiro, 1991). These measures and their corresponding formulae defined in terms of probabilities of absolute co-occurrence frequency counts of two-items are shown in Table 2.

Table 2: Association measures and their formulae

#	Measure	Formula
1	Interest (I)	$\frac{P(A,B)}{P(A)P(B)}$
2	Cosine (IS)	$\frac{P(A,B)}{\sqrt{P(A)P(B)}}$
3	Jacaard (ς)	$\frac{P(A,B)}{P(A)+P(B)-P(A,B)}$
4	Piatetsky-Shapiro (PS)	$P(A,B)-P(A)P(B)$

We compute association between 2-itemsets on the sample dataset given in the Table 1 using the metrics specified in the Table 2 to demonstrate the conflicts. The results, shown in Table 3, are sorted according to their association measure value in decreasing order of magnitude. Table 3 results show that different metrics produces different ordering of itemsets and all rankings are not one and the same for the metrics under considerations e.g., the itemset {G, I} is ranked highest by the Jaccard and PS measures but ranked second and fifteenth according to the cosine and Interest measures respectively. Thus a natural question arises, as to, which association measure is suitable to compute the proximity measure of database items? The right answer to this question will guide us towards the right ordering of itemsets that can be utilized for the discovery

of *top-most* and *top-k* maximal frequent itemsets without user provided *min-sup* threshold.

Table 3: Ranking of 2-items by different association metrics

Interest(I)		Cosine(IS)		Jaccard		PS	
KL	12.500	GI	0.858	GI	0.300	GI	0.184
LM	12.500	DF	0.770	DF	0.276	DF	0.147
MN	12.500	BC	0.740	EG	0.269	EG	0.136
KM	6.250	BF	0.740	BC	0.267	BC	0.133
NO	6.250	EG	0.738	BF	0.267	BF	0.133
AP	3.125	BD	0.721	BD	0.265	EI	0.122
HO	3.125	KL	0.707	EI	0.259	BD	0.110
JO	3.125	LM	0.707	KL	0.250	JO	0.082
MO	3.125	MN	0.707	LM	0.250	EF	0.078
OP	3.125	EI	0.704	MN	0.250	AB	0.077
AF	2.083	JO	0.612	JO	0.231	CE	0.070
HJ	2.083	CE	0.556	CE	0.217	AF	0.062
JK	2.083	DI	0.522	DI	0.207	AD	0.043
JP	2.083	CI	0.503	BI	0.200	CI	0.042
GI	2.045	BI	0.502	CI	0.200	KL	0.037
EG	1.944	AF	0.500	KM	0.200	LM	0.037
AB	1.923	KM	0.500	CD	0.192	MN	0.037
DF	1.852	NO	0.500	AB	0.190	KM	0.034
EI	1.768	CD	0.481	AF	0.188	NO	0.034
BC	1.709	CF	0.444	CF	0.182	CF	0.030
BF	1.709	AD	0.433	CG	0.174	DI	0.029
AD	1.563	CG	0.422	NO	0.167	CD	0.027
CE	1.543	DE	0.385	DE	0.160	AP	0.027
BD	1.442	BE	0.370	AD	0.158	HO	0.027
EK	1.389	DG	0.365	BE	0.154	MO	0.027
FP	1.389	AP	0.354	DG	0.154	OP	0.027
CI	1.263	HO	0.354	BG	0.148	HJ	0.021
GH	1.250	MO	0.354	AP	0.143	JK	0.021
GK	1.250	OP	0.354	HO	0.143	JP	0.021
CF	1.235	BG	0.351	MO	0.143	CG	0.016
CD	1.157	HJ	0.289	OP	0.143	EK	0.011
DI	1.136	JK	0.289	EJ	0.118	FP	0.011
HI	1.136	JP	0.289	GJ	0.111	BI	0.011
IK	1.136	EJ	0.272	HJ	0.111	GH	0.008
CG	1.111	GJ	0.258	JK	0.111	GK	0.008
BI	1.049	IJ	0.246	JP	0.111	HI	0.005
AJ	1.042	DJ	0.236	IJ	0.105	IK	0.005
DP	1.042	EK	0.236	DJ	0.100	AJ	0.002
BP	0.962	FP	0.236	EF	0.100	DP	0.002
DE	0.926	GH	0.224	AJ	0.091	BP	-0.002
EJ	0.926	GK	0.224	FI	0.091	EJ	-0.006
BE	0.855	EF	0.222	EK	0.083	DE	-0.013
DG	0.833	HI	0.213	FP	0.083	GJ	-0.016
GJ	0.833	IK	0.213	GH	0.077	AC	-0.018
BG	0.769	AJ	0.204	GK	0.077	IJ	-0.026
IJ	0.758	DP	0.204	AC	0.071	BE	-0.027
AC	0.694	FI	0.201	HI	0.071	DG	-0.032
DJ	0.694	BP	0.196	IK	0.071	DJ	-0.035
EF	0.617	AB	0.194	DP	0.067	FJ	-0.046
FI	0.505	AC	0.167	BP	0.063	BG	-0.048
FJ	0.463	FJ	0.136	FJ	0.063	FI	-0.078
BJ	0.321	BJ	0.113	BJ	0.050	BJ	-0.085
FG	0.278	FG	0.105	FG	0.050	FG	-0.104

No association measure produces the correct ordering of these itemsets in the ranking list. These conflicting results render these metrics inappropriate to be used for the association computation. Thus, there arises a need for an appropriate association measure which can lead to right order of itemsets considering the intrinsic properties of relative frequencies. An association metric which gives highest rank to the most frequent itemset and gives lowest rank to the least frequent itemset can be used to derive required MFIs using a parameter-free approach.

2.2 Content-based Image Retrieval

Recent developments in the field of multimedia technologies spawned a huge amount of multimedia data (audio, video, images) with contents as diverse as medical databases having images like MRI, X-rays, etc, national registration databases containing citizens' images and their finger prints, criminal databases maintained by security agencies for suspect tracking, and personal or family picture collections. The transformation of a common man has intensified from a passive user of multimedia data to an active producer due to low-cost multimedia capturing devices, inexpensive storage devices, and effortless hosting on the WWW. These days, one can search multimedia databases, rapidly growing in size, with extremely diverse type of visual and semantic contents. These factors are posing countless challenges to the real-world semantic image search system designers. Traditional keyword based searching and retrieval approaches do not cope well with these new challenges. Thus, content-based image retrieval (CBIR) has recently emerged as a significant research area in multimedia computing (Smeulders, Worring, Santini, Gupta, and Jain, 2000).

CBIR can be described as the task of finding images from a large image repository that match to a given query image or keyword as illustrated in Figure 5. There exist different ways to express the query. In most systems the query is defined by providing one or more example images.

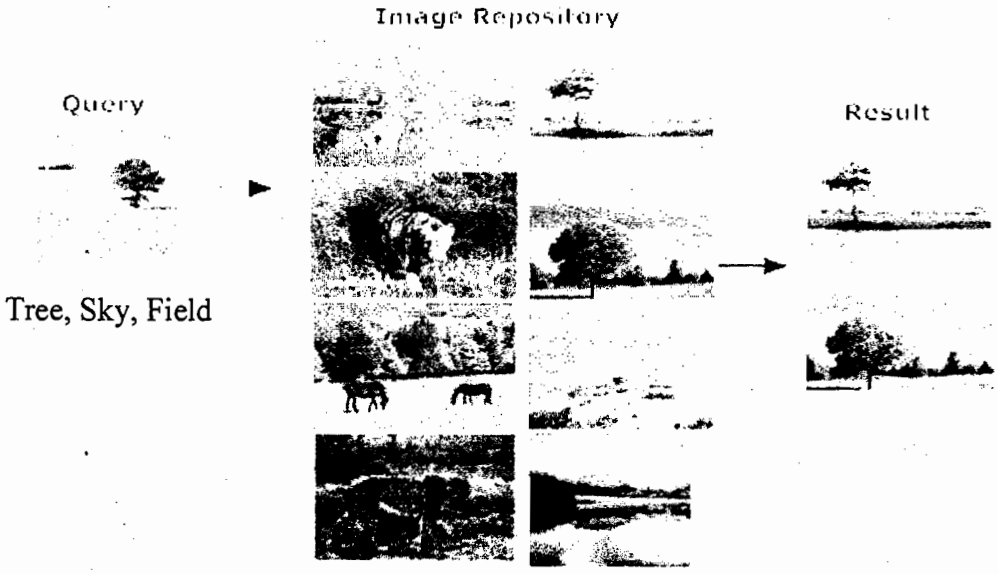


Figure 5: An illustration of content-based image retrieval

CBIR technologies permit users to organize and process multimedia repositories by their low-level visual features. Wang, Boujemaa, Bimbo, Geman, Hauptmann, and Tesic, (2006) categorize everything under the purview of CBIR ranging from an image similarity function to an image annotation system. This categorization of CBIR makes it a unique field of study among the scientific community. We also witness continuous efforts by researchers from various fields of studies such as machine learning, information retrieval, computer vision, human-computer interaction (HCI), information

theory, databases, data mining, statistics, and algorithms contributing towards solving basic image understanding issues and integrating into the CBIR community.

The CBIR systems are mainly concerned with two fundamental problems: i) which mathematical model best represents an image electronically, and ii) how to measure the visual similarity between two images quantitatively (Datta, Joshi, Li, and Wang, 2008). The *first* problem is due to the internal representation of an image as an array of pixel values. The mathematical model of an image is referred to as its *signature*. This does not correspond directly to the human perception, leaving alone understanding the image semantically. The *second* problem is about selection of visual similarity metric. A similarity metric measures the visual discriminating ability of low-level features to distinguish the different image regions. The extraction of image signatures and the computation of image similarity cannot be separated as far as system design is concerned. To a large extent the composition of similarity measures is dependent on formulation of image signatures. Often, intuitions are the early motivating factors for designing similarity measures, which in turn define requirements on the construction of signatures. For measuring visual feature similarity the most common metrics in use are *distance, information (or uncertainty), dependency, consistency*, and *classifier error rate*.

The task of semantic image retrieval can be divided in the following four steps:

i. *Segmentation and feature extraction* – The first step in image analysis framework is to parse the raw image into the major homogenous regions. The segmentation component partitions images into local contents via region based method. Then, the feature extraction component extracts low-level features from the segmented images i.e., each segmented region is represented by some visual feature vectors.

ii. *Feature dimension reduction* – The extracted image features are often high-dimensional because the model requires large number of parameters. As a post-processing step it is essential to apply a dimensions reduction methodology to remove the redundant features since feature vectors with reduced amount dimensions are more efficient.

iii. *Similarity measure* – Similarity measure is used to compute the distance between query image and the corresponding images in database. The computed distance is a measure of visual similarity of low-level contents for estimating semantic similarity.

iv. *Retrieval* – Using the system to retrieve alike images.

Existing image retrieval systems compute similarity between users provided query image feature(s) and low-level features of images stored in databases. The visual similarity is then compared against a threshold value. All images are retrieved from the database crossing the threshold. Consequently, only the images similar to the query image(s) in terms of visual content are retrieved. This similarity checking on image signatures and annotated key words using a similarity metric is very time consuming.

Image retrieval has an emerging interest in various fields of study, such as multimedia databases, information retrieval, algorithms, machine learning, and human-computer interaction (HCI). In the context of interpreting images, despite the advancement made in the field of image retrieval research, a universally acceptable algorithm to portray human perception is not available readily. Datta et al. (2008) conclude that "it is not surprising to see continued effort in this direction, either building upon prior work, or

exploring novel directions. Considerations for successful deployment of CBIR in the real-world are reflected by the research focus in this area".

2.2.1 Semantic and Sensory Gaps

Selection of right features play critical role in image classification and retrieval systems. Feature selection is closely related to the object representation. Typically images are represented in the form of feature vectors. The dimension of these feature vectors is usually very high. The high dimensionality of feature vectors poses great challenges to indexing and retrieval tasks and is known as the curse of dimensionality in the research community. The problem with most of the current approaches for judging semantic similarity between low-level image features and higher-level semantic concepts is the reliance on comparison of similarity with all the database images.

Datta et al. (2008) have quoted various *gaps* that define and motivate most of the related problems as follows:

> —The *sensory gap* is the gap between the object in the world and the information in a (computational) description derived from a recording of that scene.
>
> —The *semantic gap* is the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for a user in a given situation.

The task of visual feature extraction from images is divided into two steps, namely image processing and feature construction. The issue here is to identify what features shall be extracted to perform meaningful retrieval. Once image features are extracted, next question arises as to how they could be indexed and matched against each other.

Essentially, it is desired to reduce the semantic gap as much as possible, sometimes also reducing the sensorial gap as well in the process.

2.2.2 Formal Specification

In recent years the CBIR has remained a significant theme of research. Early retrieval systems are based on the query-by-example paradigm, realizing that the design of fully functional feature based retrieval systems would require support for semantic queries. These systems consist of a database of images annotated with semantic keywords, enabling the user to specify the query through a natural language description of the visual concepts of interest. The problem in which we are interested in can be formulated as, "*How to map frequent pattern mining framework to retrieve similar images for the best database match to a user provided query image?*" A desirable solution should be able to include all visual features, and detect similarities among any subset of images. Moreover, in terms of computation, we would like a method that is scalable to the database size and does not require parameter setting by the human. This realization, combined with the cost of manual image labeling, generated significant interest in the problem of automatically extracting semantic descriptors from images.

CHAPTER 3

RELATED WORK

In this chapter, the related work is reviewed to identify the strengths and weaknesses in existing research. Strengths provide solid foundations for the study and weaknesses help in establishing the originality of the research objectives. Topics that are reviewed include parameter-free mining, frequent patter mining, graph theoretic approaches, image feature selection, and semantic image retrieval.

3.1 Parameter-free Mining

In the context of data mining Anders and Sester (2000) were the first who proposed the concept of parameter-free mining algorithm for cluster detection in the spatial databases. Algorithms in computer science and especially in data mining abound. Majority of algorithms require a number of parameters that must be tuned. In general, parameter-laden approaches may tune their parameters in such a way to get excellent performance on one data set, but may totally fail to show similar results when applied to a new but very similar data set. During the literature survey it is observed that a number of techniques are being proposed to automatically tune parameters and many of these methodologies themselves require parameters, resulting in an infinite loop. A parameter-laden approach poses another problem when it gets difficult to replicate the published experimental results. This also makes it very hard to evaluate the contributions made by a proposed algorithm.

Keogh et al. (2004) show that employing parameter-free data mining approach based on compression can obtain good results. They propose parameter-free algorithms for a number of data mining tasks such as anomaly detection, classification, and clustering. However, no parameter-free solution is proposed for the problem of frequent pattern mining.

Recent research advances (Cheung and Fu, 2004; Wang et al., 2005) introduce another parameter called *top-k* to discover *top-k* frequent patterns. This retrieval process is accomplished without specifying *min-sup*. Cheung and Fu (2004) at first build a complete FP-tree in memory, and then derive *k most frequent l*-itemsets (where *l* is within user specified range). Various heuristics are applied to compress the result sets and search space for mining *top-k* frequent patterns. Wang et al. (2005) aimed at discovering *top-k closed* itemsets exceeding a user specified length. Under this condition, an FP-tree is built using several pruning strategies such as transactions having length less than the specified length are omitted.

More recently Chuang et al. (2008) propose mining *top-k* (MTK) algorithm for mining frequent itemsets and closed itemsets realizing the concept of the δ-stair search to retrieve *top-k* (closed) frequent patterns. MTK is developed to work without min-sup threshold by integrating the merit of the *horizontal first search* and the *vertical first search* approaches called *δ-stair search*. MTK claims to be working without specifying min-sup in advance however, it still internally maintains min-sup by initially setting the min-sup equal to zero, and then raising the min-sup equal to $sup_k(w)$ after the w^{th} database scan. MTK still suffers from multiple database scans.

3.2 Frequent Pattern Mining

The frequent pattern mining literature argues that the Apriori is the first algorithm proposed by Agrawal and Srikant, (1994) in the form of association rule mining. Many of the subsequently proposed association rule algorithms are its variant. However, the term association rules can be traced back to more general way as representation of knowledge about a relation between two Boolean attributes (created e.g. as conjunctions attribute value pairs) that are supported by the analysed data. This more general understanding of association rules is the basis of an exploratory data analysis method, called General Unary Hypothesis Automaton (GUHA), first introduced by H´ajek, Havel, and Chytil in 1966. The basic principle of GUHA method is to let the computer generate and evaluate all hypotheses that may be interesting from the point of view of the given data and the studied problem.

The Apriori (Agrawal and Srikant, 1994) algorithm lists every frequent itemset using a bottom-up breadth-first search technique. It performs well when the length of derived itemsets is small but it would require too much time if the length of resultant itemsets is large. The performance of Apriori-like algorithms' is marred by the exponential growth of candidate itemsets and repeated database scans at each iteration to check support. Later a number of algorithms using different techniques have been proposed to improve upon the efficiency of Apriori-like algorithms.

FP-growth proposed by Han and Pei (2000) is a different method, which basically adopts a different strategy from the Apriori-like algorithms. FP-growth finds frequent itemsets without candidate set generation. In first step FP-growth compresses the database into a compact structure, called FP-tree, by retaining the itemset association information to

avoid repeated database scans. It then divides such a compressed database into a set of conditional databases, each associated with one frequent item, and mines each such database separately. FP-growth show good efficiency against Apriori class of algorithms due to compressed representation of the entire database in main memory. However, it is not scalable due to the main memory requirements of the FP-trees. At the same, since it derives all frequent patterns it is not reasonable when pattern length is long.

Max-Miner (Bayardo, 1998) uses set enumeration tree to discover all frequent itemsets utilizing breadth-first traversal of the search space. This reduces database passes by employing a look-ahead pruning strategy to remove both supersets of infrequent itemsets and subsets of frequent itemsets. When the support of an itemset is computed, at the same time the support of the largest itemset that appears in the subtree rooted at this itemset is also calculated. If this superset is frequent, then all other itemsets in this subtree must be frequent too. In this way Max-Miner avoids to work out the support of all the frequent itemsets. Since, it uses the horizontal database format that causes to perform multiple passes over the database as Apriori does.

DepthProject (Agarwal, Aggarwal and Prasad, 2000) is another attempt to discover frequent patterns. This method builds a lexicographic tree of itemsets and then using depth-first search and counting based on transaction projections along its branches to derive long itemsets. DepthProject returns a superset of the MFI using look-ahead pruning with item reordering and require post-pruning strategies to eliminate non-maximal patterns.

Mafia (Burdick, Calimlim and Gehrke 2001) is one of the most efficient techniques for finding maximal frequent itemsets. Initially the data is converted into a vertical bit-vector

layout, and then uses compression and projection of bitmaps for performance enhancement. Mafia uses a number of pruning methodologies to remove non-maximal itemsets such as; i) look-ahead pruning, ii) to check if a new set is subsumed by an existing maximal set and, iii) if $t(X)$ subset $t(Y)$. Mafia mines a superset of the MFI, and requires a post-pruning step to eliminate non-maximal patterns. The most time consuming step involve is the conversion of database into vertical bit-vectors format. GenMax (Gouda and Zaki, 2005) uses backtracking based search method for mining MFIs. GenMax uses a number of strategies to prune the search space such as *progressive focusing* to perform maximality checking, and *diffset propagation* to perform fast frequency computation. While Mafia is best for discovering a *superset* of all maximal itemsets, GenMax is the algorithm of preference for enumerating the *exact* set of maximal itemsets.

Khayal, Rehman, Khan, Salam (2007) attempt to cluster FIs using Jacuard similarity measure. It make use of prefix tree as an underlying data structures on vertical database layout to cluster frequent itemsets. The experimental results show that FIs are generated successfully but still require *min-sup* to be adjusted by the user.

3.3 Graph Theoretic Approaches

Majority of frequent pattern generation methods generally use tree based data structures. The most common data structures types used are hash trees, set-enumeration trees, FP-tree, and prefix trees also known as tries. Little work is reported in the literature using graph data structure. DLG (Yen and Chen, 1996) is a first graph-theoretic approach developed for finding frequent itemsets. The database is initially converted to a lexicographic vertical bit vector format. Infrequent items are removed using bit vector

summation, resulting in V_1. An association graph is then constructed from V_1 where an edge between items i and j: $i < j$, $|i \cap j| \geq minsup$, producing V_2. Subsequent V_κ, $\kappa > 2$ generation is based on direct extension where the edges from the last node of $V_{i\kappa}$, are used to create $C_{\kappa+1}$. If no edges exist, then no $\kappa+1$ itemsets based upon it can exist. The algorithm is very costly first due to conversion of original data into bit-vector format, second performing logical AND operations on bit vectors.

DLG* (Lee, Lee and Chen, 2001) is revised version of DLG that improves upon DLG by reducing the number of candidates and the cost of performing logical AND operations on bit vectors. It extends each large k-itemset in L_k ($k \geq 2$) into (k + 1)-itemsets, if the (k + 1)-itemset $\{i_1, i_2, ..., i_k, i\}$ is large, it must satisfy the two conditions, i) Any $\{i_j, i\}$ ($1 \leq j \leq k$) must be large, ii) and a directed edge from i_k to item i means that $\{i_j, i\}$is also a large 2-itemset. These, checks significantly reduce the number of candidate itemsets. However, DLG and DLG* incur additional cost of converting transaction database into bit vectors that makes the logical AND operation slow.

MFSET (Liu and Pan, 2007) is the most recent graph based maximal frequent pattern mining method. All MFIs in a database are derived using breadth-first search and depth-first search techniques. However, MFSET requires two database scans. DLG, DLG* and MFSET all employ directed weighted graphs. Contingency matrix is used to implement directed graph that results in wastage of memory.

3.4 Content-based Image Retrieval

Modern technological advancements in the fields of computer, digital still cameras and digital video cameras, and the Internet have led to the collection of large volumes of image data. Due to this exponential growth in digital image libraries it is required to

systematize image collections in a meaningful way. The need for semantic image retrieval has become increasingly important. Typically content-based image retrieval (CBIR) systems are founded on quantifying the similarity between the user provided image feature(s) and the low-level extracted features of images stored in the database. As a result, the derived images are similar to query image in terms of visual content.

The literature of digital image processing and data mining is rich as far as CBIR systems are concerned. QBIC (Flickner et al., 1995) is the first image retrieval system in the mid-1990s. Other renowned research systems are VisualSEEk (Smith and Chang, 1996), Pictoseek (Gevers and Smeulders, 2000), SIMPLIcity (Wang, Li and Wiederhold, 2001), BlobWorld (Carson, Belongie, Greenspan and Malik, 2002), Virage (Gupta and Jain, 1997), and CHROMA (Lai, 2000). However, retrieval results of majority of the existing CBIR systems are not satisfactory. This is because human perception of images is based upon high-level semantic concepts and users pose natural language-like queries, i.e., typically query images by semantics. Image annotation or automatically annotate images with keywords is a solution to this problem. Several image annotation systems have also been proposed in the literatures. Image annotation can be perceived as a process of translating "visual language" to text. They utilize a machine translation model to collect the links between the words and visual features, and use these links to annotate new image. Jeon, Lavrenko and Manmatha (2003) propose cross media relevance model to approximate the combined probability of the image and the labels. Monay and GaticaPerez (2003, 2004) propose latent semantic analysis (LSA) and probabilistic latent semantic analysis (PLSA) respectively using latent variables to link image features with keywords. Text classification techniques can also be applied to image annotation task by

taking each annotation word as a separate category, and creating a different classification model for each word. Using classification technique Gao, Fan, Xue and Jain (2006) introduce a multi-resolution grid-based annotation method for semantic image representation and a hierarchical boosting algorithm to address the image annotation problem. A fully automatic and high speed annotation system–ALIPR is constructed by Li and Wang (2006). Some previous works demonstrate that utilizing keyword correlations can improve upon the performance of image annotation. A subset of the annotation vocabulary can be used to exploit the keyword correlations to get better image annotations. However, for a large size vocabulary, it is computationally difficult to enumerate all the subsets of keyword. Jin, Chai and Si (2004) address annotation problem by using EM-algorithm to apply a language model to generate an annotation subset. However, the speed remains slow due to the EM-algorithm. Srikanth, Varner, Bowden and Moldovan (2005) propose a hierarchical classification approach to image annotation. They use a hierarchy induced on the annotation words derived from WordNet (Miller, Beckwith, Fellbaum, Gross and Miller, 1990).

Baeza-Yates and Ribeiro-Neto (1999) make use of the knowledge-based WordNet and multiple evidence combination to prune irrelevant keywords. Liu, Li, Ma, Liu and Lu (2006) propose an adaptive graph model based on manifold ranking, which combines the WordNet and the pair-wise keyword co-occurrence to improve the annotation effectiveness.

From the perspective of content-based image retrieval, two widely accepted frameworks used in image annotation are: i) automatic annotation, and ii) manual annotation. Automatic image annotation systems derive semantics using a statistical or machine

learning model that initially train the learning method on a set of annotated images and then uses this model to relate visual features of fragmented regions with semantic concepts. However, developing practical automatic image annotation methods are still a far away idea. Relevance feedback (manual annotation) is proposed as an alternative framework to reduce the semantic gap by incorporating human perception called user feedback early in the image annotation process.

Majority of image retrieval systems do feature extraction as an initial step. Once extracted these visual features are then used as inputs to subsequent tasks. Recent research has shown great promise in region-based visual features, for which segmentation is the quite essential first step (Tsai and Hung, 2008).

3.4.1 Image Segmentation

Image segmentation is a key step in acquiring region-based signatures from raw images. Efficient segmentation method is vital for distinguishing and estimating object shapes within images. Segmentation divides the whole image region into salient object regions making extraction of low-level image features simple. Tsai and Hung (2008) classify segmentation strategies into two categories; the first approach divides an image into fixed number (like 2, 4, or 5) and size of blocks or tiles. The second approach divides an image into a number of variable shaped object regions of interest. After performing block based or region based segmentation, low-level pixel feature(s) are then extracted from the tiles or regions for local feature representation.

Saux and Amato (2004) argue that meaningfully region-based segmented images can be used more efficiently to provide added semantic information than using global image features. Hence, images containing same types of segments are more likely to be

associated with a particular semantic concept. To obtain a compact representation of the image features Loeff , Alm and Forsyth (2006) extract patches automatically around interesting regions called *keypoints* in each image. Since human perception is often concentrated only to a part or an object of an image therefore majority of methodologies advocate focusing on local characteristics for capturing low-level information to be used for retrieval or annotation.

3.4.2 Features Extraction

Majority of the CBIR systems compare images based on low-level features in one way or another and therefore a large variety of features for image retrieval and annotation exist. Different approaches are employed to extract low-level visual features based on the image constituent parts. Generally, all known features are not utilized by the CBIR systems because this would involve huge amounts of data and increase the processing time significantly. Instead, only a set of some important visual features are usually selected. The most commonly used visual content descriptors are color, texture, and shape.

a) Color

It is understood that the human eye perceives color features very well and very quickly. Image color descriptors can be organized in different manners such as *color histogram, color correlogram, color coherence vector, and color moments*. The *color histogram* provides an efficient representation of the color content provided the color pattern is unique compared with the rest of database. It is straightforward to calculate the color histogram and effective in representing both the global and local distribution of colors in an image. Further, the color histogram is robust to translation and rotation about the view

axis and changes only slowly with the scale, occlusion, and viewing angle. However, the color histogram captures only the global color features of the images and does not contain spatial information among image colors. To incorporate spatial information into the color histogram Pass and Zabith (1996) proposed a different framework called *color coherence vectors* (*CCV*). In this framework each histogram bin is divided into either *coherent*, if it belongs to a uniformly-colored region, or *incoherent*, if it does not. Huang et al. (1997) propose the *color correlogram* to differentiate between not only the local color distributions of pixels, but also the global distribution of this spatial correlation of pairs of colors. A color correlogram is a table indexed by color pairs, where $k_{(i,j)}$ represents the probability of finding a pixel of color j at a distance k from a pixel of color i in the image. The accuracy of the image representation increases and false positives of the image retrieval systems decreases when spatial information is integrated into the color features.

Histogram based techniques are liable to fail when two images with similar color histograms have a very different spatial appearance. *Color moments* have been successfully used especially when the image contains just objects. Usually the performance of color moment is good if it is defined by both the L*u*v* and L*a*b* color spaces as opposed to solely by the HSV space (Long, Zhang and Feng, 2006).

b) Texture

Texture features represent an image's characteristics like the *regularity, coarseness,* and *smoothness* of visual perception. Long, Zhang and Feng (2006) classify texture representation methods into two categories; i) *structural* and, ii) *statistical*. Structural methods express texture by recognizing structural primitives and their placement rules.

This strategy is very effective for regularly repeated type of textures. Alternatively, statistical methods describe texture by the statistical distribution of the image intensity.

c) Shape

Many content-based image retrieval systems extract object boundaries to be used as another important image feature. These object boundaries are recognized as *shape* features. Usually, image segmentation is a pre-processing step to derive shape features. The application of shape features for image retrieval is restricted to the cases where objects or regions are distinctly obtainable because robust and precise segmentation is very difficult to realize. The image shape extraction techniques can be categorized into either *boundary-based* or *region-based* methods. A good shape representation feature for an object should be invariant to translation, rotation, and scaling.

3.4.3 Discussion

Majority of existing semantic image retrieval approaches extract low-level features from image regions and build a model between images and keywords based on a reference captioned dataset. However, most of the training datasets are constructed by a special interest group to solve a specific problem like object class recognition (motorbike, car, bicycle, and cow) or object instance recognition such as face detection. Images in these datasets are labeled manually without extracting any information of the relationships between segmented regions and captions. The quality of image labeling is also not of high quality. Many datasets provide object labels without specifying the location of the object in the image. However, more detailed information, if available, is very helpful for semantic image processing.

The performance comparison of different techniques is not an easy task. Majority of previous works do not report results on the same datasets as standards for image datasets have not been established. Further, various accuracy measures like precision, recall, and mean average precision have been used to compute efficiency. Since the human perception of an image is subjective, some works also report user evaluation of the annotation result. This establishes the need for further research to address these issues.

CHAPTER 4

MINING WITHOUT MINIMUM SUPPORT THRESHOLD

In the previous chapter the conventional frequent pattern mining methods were reviewed. The major drawbacks of these methods are;

- Huge number of candidate generation and candidate testing
- Maintenance of support counts for each candidate sets
- Multiple database scans
- Large number of resulting itemsets, and
- To tune the *min-sup* over a wide range of values.

Majority of existing frequent pattern mining approaches mostly concentrate on improving search time by employing complex data structures, efficient search techniques and pruning strategies. In this work, a novel approach is presented that makes use of graph theoretic data structure to effectively retrieve maximal frequent patterns. The prominent advantage of this approach is that it does not require user specified *min-sup* threshold and performs single database scan only. This technique can be applied to retrieve the *top-most* and *top-k* maximal frequent itemsets. In case of mining more than one maximal frequent itemset, the algorithm requires a more easily understandable parameter k, which refers to the desired number of maximal frequent itemsets.

First, an association metric called *association ratio* for measuring the proximity distance between the items is introduced in Section 4.1. Then, in Section 4.2, we present an undirected weighted graph structure called *association ratio graph*. In Section 4.3, we propose efficient algorithms for mining *top-most* and *top-k* maximal frequent itemsets using association ratio graph, and show its properties and correctness. Experimental studies and performance results are discussed in the Section 4.4.

4.1 Association Ratio Metric

A number of metrics have been proposed in the literature to compute an association between items in a transaction database. Pang-Ning, Vipin and Jaideep (2004) present a comprehensive overview of the proposed association measures and discuss many important characteristics one should consider before selecting a metric to be used for a given application domain. A complete list of different measures proposed in the fields of machine learning, statistics and data mining are given in Table 4. Pang-Ning et al. (2004) have divided the measures given in Table 4 into two types using *symmetry under variable permutation* property: i) asymmetric measures, and ii) symmetric measures. Most measures of association are symmetric. It means that the value of the measure is the same which ever variable is considered to be the dependent variable. Whereas, the asymmetric measures have one value if the row variable is the dependent variable, and another if the column variable is dependent. Examples of asymmetric measures are J-Measure, added value, laplace, confidence, conviction, mutual information, gini index, and Klosgen.

Table 4: List of interestingness measures adopted from Pang-Ning et al. (2004)

#	Measure	Definition
1	ϕ-coefficient	$\frac{P(A,B)-P(A)P(B)}{\sqrt{P(A)P(B)(1-P(A))(1-P(B))}}$
2	Goodman-Kruskal's (λ)	$\frac{\sum_j \max_k P(A_j,B_k)+\sum_k \max_j P(A_j,B_k)-\max_j P(A_j)-max_k P(B_k)}{2-\max_j P(A_j)-\max_k P(B_k)}$
3	Odds ratio (α)	$\frac{P(A,B)P(\overline{A},\overline{B})}{P(A,\overline{B})P(\overline{A},B)}$
4	Yule's Q	$\frac{P(A,B)P(\overline{A}\overline{B})-P(A,\overline{B})P(\overline{A},B)}{P(A,B)P(\overline{A}\overline{B})+P(A,\overline{B})P(\overline{A},B)} = \frac{\alpha-1}{\alpha+1}$
5	Yule's Y	$\frac{\sqrt{P(A,B)P(\overline{A}\overline{B})}-\sqrt{P(A,\overline{B})P(\overline{A},B)}}{\sqrt{P(A,B)P(\overline{A}\overline{B})}+\sqrt{P(A,\overline{B})P(\overline{A},B)}} = \frac{\sqrt{\alpha}-1}{\sqrt{\alpha}+1}$
6	Kappa (κ)	$\frac{P(A,B)+P(\overline{A},\overline{B})-P(A)P(B)-P(\overline{A})P(\overline{B})}{1-P(A)P(B)-P(\overline{A})P(\overline{B})}$
7	Mutual Information (M)	$\frac{\sum_i \sum_j P(A_i,B_j)\log\frac{P(A_i,B_j)}{P(A_i)P(B_j)}}{\min(-\sum_i P(A_i)\log P(A_i), -\sum_j P(B_j)\log P(B_j))}$
8	J-Measure (J)	$\max\left(P(A,B)\log(\frac{P(B\|A)}{P(B)}) + P(A\overline{B})\log(\frac{P(\overline{B}\|A)}{P(\overline{B})}),\right.$ $\left. P(A,B)\log(\frac{P(A\|B)}{P(A)}) + P(\overline{A}B)\log(\frac{P(\overline{A}\|B)}{P(\overline{A})})\right)$
9	Gini index (G)	$\max\left(P(A)[P(B\|A)^2 + P(\overline{B}\|A)^2] + P(\overline{A})[P(B\|\overline{A})^2 + P(\overline{B}\|\overline{A})^2]\right.$ $-P(B)^2 - P(\overline{B})^2,$ $P(B)[P(A\|B)^2 + P(\overline{A}\|B)^2] + P(\overline{B})[P(A\|\overline{B})^2 + P(\overline{A}\|\overline{B})^2]$ $\left.-P(A)^2 - P(\overline{A})^2\right)$
10	Support (s)	$P(A,B)$
11	Confidence (c)	$\max(P(B\|A), P(A\|B))$
12	Laplace (L)	$\max\left(\frac{NP(A,B)+1}{NP(A)+2}, \frac{NP(A,B)+1}{NP(\overline{B})+2}\right)$
13	Conviction (V)	$\max\left(\frac{P(A)P(\overline{B})}{P(A\overline{B})}, \frac{P(B)P(\overline{A})}{P(B\overline{A})}\right)$
14	Interest (I)	$\frac{P(A,B)}{P(A)P(B)}$
15	cosine (IS)	$\frac{P(A,B)}{\sqrt{P(A)P(B)}}$
16	Piatetsky-Shapiro's (PS)	$P(A,B) - P(A)P(B)$
17	Certainty factor (F)	$\max\left(\frac{P(B\|A)-P(B)}{1-P(B)}, \frac{P(A\|B)-P(A)}{1-P(A)}\right)$
18	Added Value (AV)	$\max(P(B\|A) - P(B), P(A\|B) - P(A))$
19	Collective strength (S)	$\frac{P(A,B)+P(\overline{A}\overline{B})}{P(A)P(B)+P(\overline{A})P(\overline{B})} \times \frac{1-P(A)P(B)-P(\overline{A})P(\overline{B})}{1-P(A,B)-P(\overline{A}\overline{B})}$
20	Jaccard (ζ)	$\frac{P(A,B)}{P(A)+P(B)-P(A,B)}$
21	Klosgen (K)	$\sqrt{P(A,B)}\max(P(B\|A) - P(B), P(A\|B) - P(A))$

In data mining practices, asymmetric metrics are applied to measure the statistical significance of derived implication rules, where it is required to differentiate between the strength of the rule $A \rightarrow B$ from $B \rightarrow A$, whereas, symmetric measures such as cosine (IS), Φ-coefficient, interest factor (I), Jaccard (ζ) and odds ratio (α) are appropriate to measure the interestingness among the database transactions items. The symmetric measures disregard the transaction records that do not contain the two items under consideration. This characteristic of symmetric measures is useful for domains where co-occurrence of two items is more important than their co-absence. However, existing metrics give contradictory information about the interestingness of a pattern, and the best measure to use is hardly known. No such interestingness measure is known that is always better than others in all application areas. This is due to the diversified characteristics of each metric introduced for specific application domains.

Association metrics capture the dependency between the two items by computing a statistic. The strength of the association is indicated by the absolute value of the metric. Larger values indicate strong association and greater likelihood of items being part of frequent itemsets and poorly correlated and uncorrelated items are eliminated by assigning a value zero. A contingency table can show how strong the relationship is? Symmetric measures are defined in terms of the probabilities (relative frequency counts) computed over a 2-way contingency table as shown in Table 5.

Table 5: A two-way contingency table

	B	$\overline{B}$
A	P(A, B)	$P(A, \overline{B})$
$\overline{A}$	$P(\overline{A}, B)$	$P(\overline{A}, \overline{B})$

Numerous famous symmetric association measures such as interest factor (I) (Brin et al., 1997), cosine (IS) (Tan and Kumar, 2000), Jaccard (ζ) (Van Rijsbergen, 1979), Piatetsky-Shapiro (PS) (Piatetsky-Shapiro, 1991) and odds ratio (α) along with their corresponding formulae are shown in Table 6 for computing similarity between a pair of items.

Table 6: Symmetric association measures

	Measure	Formula
1.	**Interest (I)**	$P(A,B)/P(A)P(B)$
2.	**Cosine (IS)**	$P(A,B)/\sqrt{P(A)P(B)}$
3.	**Piatetsky-Shapiro (PS)**	$P(A,B)-P(A)P(B)$
4.	**Jaccard (ξ)**	$P(A,B)/(P(A)+P(B)-P(A,B))$
5.	**Odds ratio (α)**	$P(A,B)P(\overline{A},\overline{B})/P(A,\overline{B})P(\overline{A},B)$

Initially the example database is scanned and all pairs of 2-itemset are generated. We now illustrate the computation of *association* for the 2-itemsets using the metrics given in Table 6 on the sample dataset of Table 1. The associations computed using the interest, cosine, Jacaard, Piatetsky-Shapiro and odds ratio are shown in the first five columns of Table 7 respectively. The results are ranked according to the decreasing order of association magnitude. As can be seen the rankings are not identical for all the measures. Different metrics guide towards considerably different rankings of 2-itemsets.

For example, consider some poorly correlated itemsets such as KL, LM, MN, KM, NO and OP. These itemsets are marked grey in the Table 7 indicating the itemsets were not ranked properly. These itemsets were ranked highest by the interest (I) and odds ratio (α) measures, whereas, these poorly correlated itemsets must appear at the bottom of the ordered list. This was because their relative co-occurrence frequency counts were very

small in the sample database. These itemsets emerged at various positions in the rankings due to the five measures.

Table 7: Computation of association ratio using metrics of Table 6

Interest (I)		Cosine (IS)		Jacaard ()		PS		Odds		AR	
KL	12.500	GI	0.868	GI	0.750	GI	0.184	AB	INF	BD	0.563
LM	12.500	DF	0.770	DF	0.615	DF	0.147	KL	INF	GI	0.563
MN	12.500	BC	0.740	EG	0.583	EG	0.136	LM	INF	BC	0.471
KM	6.250	BF	0.740	BC	0.571	BC	0.133	GI	58.50	BF	0.471
NO	6.250	EG	0.738	BF	0.571	BF	0.133	DF	24.00	DF	0.471
AP	3.125	BD	0.721	BD	0.563	EI	0.122	KM	22.00	EG	0.389
HO	3.125	KL	0.707	EI	0.538	BD	0.110	MN	22.00	EI	0.389
JO	3.125	LM	0.707	KL	0.500	JO	0.082	JO	18.00	BI	0.316
MO	3.125	MN	0.707	LM	0.500	EF	0.078	BC	17.60	DI	0.316
OP	3.125	EI	0.704	MN	0.500	AB	0.077	BF	17.60	CD	0.250
AF	2.083	JO	0.612	JO	0.429	CE	0.070	EG	15.17	CE	0.250
HJ	2.083	CE	0.556	CE	0.385	AF	0.062	EI	10.50	CI	0.250
JK	2.083	DI	0.522	DI	0.363	AD	0.043	AF	7.50	AB	0.190
JP	2.083	CI	0.503	KM	0.333	CI	0.042	BD	6.75	BE	0.190
GI	2.045	BI	0.502	BI	0.333	KL	0.037	AP	6.67	BG	0.190
EG	1.944	AF	0.500	CI	0.333	LM	0.037	HO	6.67	CF	0.190
AB	1.923	KM	0.500	CD	0.313	MN	0.037	MO	6.67	CG	0.190
DF	1.852	NO	0.500	AB	0.308	KM	0.034	NO	6.67	DE	0.190
EI	1.768	CD	0.481	AF	0.300	NO	0.034	OP	6.67	DG	0.190
BC	1.709	CF	0.444	CF	0.286	CF	0.030	AD	4.00	AD	0.136
BF	1.709	AD	0.433	CG	0.267	DI	0.029	CE	3.75	AF	0.136
AD	1.563	CG	0.422	NO	0.250	CD	0.027	HJ	3.60	JO	0.136
CE	1.543	DE	0.385	DE	0.235	AP	0.027	JK	3.60	DJ	0.087
BD	1.442	BE	0.370	AD	0.231	HO	0.027	JP	3.60	EF	0.087
EK	1.389	DG	0.365	BE	0.222	MO	0.027	CI	2.08	EJ	0.087
FP	1.389	AP	0.354	DG	0.222	OP	0.027	EK	1.88	FI	0.087
CI	1.263	HO	0.354	BG	0.211	HJ	0.021	FP	1.88	GJ	0.087
GH	1.250	MO	0.354	AP	0.200	JK	0.021	CF	1.76	IJ	0.087
GK	1.250	OP	0.354	HO	0.200	JP	0.021	CD	1.61	AC	0.042
CF	1.235	BG	0.351	MO	0.200	CG	0.016	DI	1.60	AJ	0.042
CD	1.157	HJ	0.289	OP	0.200	EK	0.011	GH	1.56	AP	0.042
DI	1.136	JK	0.289	EJ	0.154	FP	0.011	GK	1.56	BJ	0.042
HI	1.136	JP	0.289	GJ	0.143	BI	0.011	CG	1.33	BP	0.042
IK	1.136	EJ	0.272	HJ	0.143	GH	0.008	HI	1.30	DP	0.042
CG	1.111	GJ	0.258	JK	0.143	GK	0.008	IK	1.30	EK	0.042
BI	1.049	IJ	0.246	JP	0.143	HI	0.005	BI	1.20	FG	0.042
AJ	1.042	DJ	0.236	IJ	0.133	IK	0.005	DP	1.09	FJ	0.042
DP	1.042	EK	0.236	DJ	0.125	AJ	0.002	GJ	1.00	FP	0.042
BP	0.962	FP	0.236	EF	0.125	DP	0.002	BP	0.92	GH	0.042
DE	0.926	GH	0.224	AJ	0.111	BP	-0.002	EJ	0.86	GK	0.042
EJ	0.926	GK	0.224	FI	0.111	EJ	-0.006	DE	0.80	HI	0.042
BE	0.855	EF	0.222	EK	0.100	DE	-0.013	BE	0.62	HJ	0.042
DG	0.833	HI	0.213	FP	0.100	GJ	-0.016	DG	0.58	HO	0.042
GJ	0.833	IK	0.213	GH	0.091	AC	-0.018	IJ	0.56	IK	0.042
BG	0.769	AJ	0.204	GK	0.091	IJ	-0.026	AC	0.54	JK	0.042
IJ	0.758	DP	0.204	AC	0.083	BE	-0.027	DJ	0.45	JP	0.042
AC	0.694	FI	0.201	HI	0.083	DG	-0.032	BG	0.44	KL	0.042
DJ	0.694	BP	0.196	IK	0.083	DJ	-0.035	EF	0.37	KM	0.042
EF	0.617	AB	0.194	DP	0.077	FJ	-0.046	FJ	0.28	LM	0.042
FI	0.505	AC	0.167	BP	0.071	BG	-0.048	FI	0.22	MN	0.042
FJ	0.463	FJ	0.136	FJ	0.071	FI	-0.078	BJ	0.12	MO	0.042
BJ	0.321	BJ	0.113	BJ	0.056	BJ	-0.085	AJ	0.10	NO	0.042
FG	0.278	FG	0.105	FG	0.056	FG	-0.104	FG	0.10	OP	0.042

This is because interest (I) and odds ratio (α) are sensitive to an individual item's relative frequencies. For items having low relative frequencies these measures yield high association values. Cosine, Jacaard, and Piatetsky-Shapiro could not rank these itemsets properly i.e., at the bottom of the list.

Having a right association measure is imperative and one has to understand the intrinsic properties of the existing measure. However, the study of properties of symmetric association measures does not help in the selection of right measure because all these measures do not exhibit similar properties. This establishes the fact that none of the existing symmetric measures produces effective ranking of itemsets.

These conflicting results have necessitated the introduction of a novel association metric called *association ratio* (AR). We now have Definition 7 below:

Definition 7 (association ratio (**AR**)) The *association ratio* between two items is an odds ratio, computed as $AR_{(i,j)} = P(x_i, x_j)/(1 - P(x_i, x_j))$. The probability of co-occurrence frequency of x_i and x_j is the relative frequency computed as $P(x_i, x_j) = f(x_i, x_j)/|D|$, where x_i and x_j are statistically independent and $f(x_i, x_j)$ being the frequency of the occurrence of x_i and x_j together.

The column six of Table 7 indicates associations computed using association ratio (AR) measure. AR measure validates the association between highly correlated items and nullifies the least correlated items. This can be observed from Table 7 that AR measure rightly eliminates the poorly correlated itemset by placing them at the bottom of the ordering. Similarly, AR measure also ranks the frequent itemsets high in the ranking

order. In view of Definition 7 the maximal frequent itemset (MFI) can now be redefined as follows:

Definition 8 (maximal frequent itemsets (**MFI**)) A pattern α is a *maximal frequent pattern* in set D if there exists no superset β having association ratio greater than pattern α, i.e., AR(α) > AR(β) in D.

The advantage of computation of association using AR measure is eliminating the computation of $P(A,\overline{B})$, $P(\overline{A},B)$ and $P(\overline{A},\overline{B})$ terms. The denominator can be directly computed by subtracting $P(A, B)$ from one.

4.2 Association Ratio Graph

Basically a transaction database has to be accessed for mining frequent patterns. Compact structures such as, directed graph (Lee, Lee and Chen, 2001; Liu and Pan, 2007), FP-tree (Han and Pei, 2000) and set enumeration trees are used for the concise representation of underlying data. Motivated by these ideas, a novel compressed data structure, called *association ratio graph* (AR-graph), can be developed for the underlying transaction database on the basis of the following observations:

1. Frequent patterns are composed of frequent items and frequent items regularly appear together in database transactions. This data can be transformed into a weighted graph data structure $G = (V, E)$ where set V (vertices) corresponds to database items and set E (edges) corresponds to the combine occurrence of end point vertices in database transactions respectively.

2. The association between two items can be represented as an edge weight. The association value will be larger when both items appear mutually in large number of transactions. If several transactions share a frequent 2-itemset, it may be possible to represent these as the number of occurrences recorded as frequency counts.
3. When 2-itemsets and their frequency counts can be incorporated in some compressed structure, this may eliminate the need for the repeated scans of transaction database.
4. If the co-occurrence frequency counts of 2-itemsets can be represented as associations using some association metric, the need for the generation of larger candidate itemsets may by avoided while discovering desired frequent itemsets.

With the above observations, we can construct an association ratio graph (AR-graph) as follows.

1. While scanning each transaction of the database, the co-occurrence frequency counts of 2-itemset are maintained as an edge weight in a symmetric contingency matrix representing an undirected weighted graph. The number of operations performed for the length t transaction is $\binom{t}{2}$. Total number of operations for computing association ratio is $T = \binom{t_1}{2} + \binom{t_2}{2} + \ldots + \binom{t_n}{2} = \sum_{i=1}^{n} \binom{t_i}{2}$. Where t_1, t_2, …, t_n are length of transaction and n is size of D.
2. Edge weights can then be updated using *association ratio* (AR) measure.
3. To facilitate graph traversal all edges are sorted in decreasing order of edge weight magnitude.

Based on these facts, an *association ratio graph* (AR-graph) can be defined as follows.

Definition 9 (association ratio graph (***AR-graph***)) Let D be a database of T transactions over a set of items I. A graph $G = (V, E)$ consists of a set of vertices $V = \{v_i = i \mid i \in I\}$ and a set of undirected edges $E = \{e_{i,j} \mid$ *edge between vertices* v_i *and* $v_j \in V$, v_i and v_j appear in the same transaction$\}$. Additionally, every edge has some weight $w(e_{i,j}) = AR_{(i,j)}$. The resulting graph is called an *association ratio graph*.

Each database item i corresponds to an AR-graph vertex v_i and each edge $e_{i,j}$ has some positive weight denoted by $w(e_{i,j})$. The edge weight signifies the level of association between the two vertices corresponding to the database items. Zero edge weight represents the lack of association between the two vertices. Given a vertex numbering and the edge weights, AR-graph can be represented by an adjacency matrix. Only one database scan is enough to construct the adjacency matrix. The adjacency symmetric matrix $A_{i,j}$ of the AR-graph is defined as;

$$A_{i,j} = \begin{cases} w(e_{i,j}), & \text{if } e_{i,j} \text{ exists} \\ 0, & \text{otherwise} \end{cases}$$

Initially, each edge weight $w(e_{i,j})$ is computed as relative frequencies using $P(x_i) = f(x_i)/|D|$ and $P(x_i, x_j) = f(x_i, x_j)/|D|$ measures respectively. Based on the definition of AR-graph and association ratio metric, we then have the following AR-graph construction algorithm.

/* **Algorithm 1**: Construction of association ratio graph */

Input: *Dataset D*
Output: *AR-Graph G=(V, E, w)*

```
1. Set V(G)= Φ and E(G) = Φ
2. for each transaction T1...Tk ∈ D {
3.     for i = 1 to n {                          /* where n= |Ti| */
4.         if (vi ∉ V(G)) {
5.             V(G)=V(G) ∪ vi
6.         }
7.         for j = i+1 to n {
8.             if vj ∉ V(G) {
9.                 V(G)=V(G) ∪ vj
10.            }
11.            if ei,j ∉ E(G) {
12.                E(G) = E(G) ∪ ei,j
13.            }
14.            w(ei,j)++
15.        }
16.    }
17. }
18. for each ei,j ∈ E(G) {
19.    w(ei,j) = P(ei,j)/(1-P(ei,j))          /* where P(ei,j)=f(xi, xj)/|D| */
20. }
```

Analysis. This algorithm scans the database only once. The AR-graph is built incrementally by adding new vertices and edges to the graph while scanning each database transaction. A symmetric matrix can be used to store the graph as a simple data structure. Each graph vertex corresponds to items in the database and a vertex corresponding item i is denoted as v_i. For each occurrence of item i the vertex's weight is incremented and for each co-occurrence of item i and j in a transaction an undirected edge between v_i and v_j is created. The cost of inserting a transaction t_i into an AR-graph is $O\left(\binom{n}{2}\right)$, where n is the number of items in transaction t_i and total cost of operations is $O\left(m\binom{n}{2}\right)$, where m is the weight of D. Each edge weight is incremented for each

subsequent occurrence of the items *i* and *j*. The AR-graph for Table 7 is shown as a contingency matrix in Figure 6.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
B	0.190														
C	0.042	0.471													
D	0.136	0.563	0.250												
E	-	0.190	0.250	0.190											
F	0.136	0.471	0.190	0.471	0.087										
G	-	0.190	0.190	0.190	0.389	0.042									
H	-	-	-	-	-	-	0.042								
I	-	0.316	0.250	0.316	0.389	0.087	0.563	0.042							
J	0.042	0.042	-	0.087	0.087	0.042	0.087	0.042	0.087						
K	-	-	-	-	0.042	-	0.042	-	0.042	0.042					
L	-	-	-	-	-	-	-	-	-	-	0.042				
M	-	-	-	-	-	-	-	-	-	-	0.042	0.042			
N	-	-	-	-	-	-	-	-	-	-	-	-	0.042		
O	-	-	-	-	-	-	-	0.042	-	0.136	-	-	-	0.042	
P	0.042	0.042	-	0.042	-	0.042	-	-	-	0.042	-	-	-	-	0.042

Figure 6: AR-graph adjacency matrix

A value in the square of the contingency matrix of Figure 6 indicates association ratio among the pair of corresponding items and the blank square indicates pair of items having no association.

4.2.1 Properties of AR-graph

A number of important properties can be observed from the AR-graph while being constructed from transaction database. Three cases can be considered for the database.

1. Let the maximum length of all database transactions be one. AR-graph obtained for such type of database contains only isolated vertices. Since no two items co-exists in any transaction therefore no edges are drawn among the vertices. Such type of an AR-graph does not contain any cycle. Identifying frequent itemset

from such a graph is trivial, i.e., an item having maximum frequency is the maximal frequent itemset.

2. Let the maximum length of all database transactions be two. Two further cases can be identified;
    a. Both items of a transaction do not occur in any transaction. Only one item exists in any other transaction and the other item only specifically exists in that particular transaction. Such type of data, when drawn, yields a tree and it is understood trees do not have any cycle. Thus identifying frequent itemsets using maximal cycle is not possible.
    b. In this case both items of a transaction can appear in any other transaction. Thus any item can occur with any other item in any other transaction. This type of data can perfectly be converted into an undirected weighted graph.
3. The rest of cases where transactions length is greater than two an AR-graph can be constructed easily. Since it is weighted undirected graph and a graph contains at least one cycle therefore graph items corresponding graph cycle represent maximal frequent itemset. In case of more than one sub-graph cycles the maximal cycle (having maximum edge weights) represents maximal frequent itemsets.

Property 1: A *maximal sub-graph* is a graph (containing at least three vertices) constructed from an AR-graph. At each step a maximum weight edge from AR-graph that has not yet been considered is selected along with its end point vertices and is added to the maximal sub-graph. After addition of each edge the sub-graph is checked for the

formation of one or more circuits (cycle) involving the edge added in the last. This process terminates when the required number of cycles are identified.

According to the above property v_i and v_j are the end point vertices of a selected edge e_{ij}. Since edges are selected in non-increasing order from an AR-graph therefore a cycle is formed when both end point vertices are already found in the maximal sub-graph. When the transaction database contains maximal frequent itemset of length three or greater, the vertices correspond to the items form a circuit in the maximal sub-graph.

Property 2: Suppose $\{i_1, i_2, \ldots, i_k\}$ is a maximal frequent itemset then there are weighted edges between each pair of items i_j and i_k in an AR-graph.

It is this property that holds when the itemset $\{i_1, i_2, \ldots, i_k\}$ is maximal, it must satisfy the following conditions otherwise, it cannot be maximal frequent itemset.
An edge from i_j to item i_k means that $\{v_j, v_k\}$ is also frequent 2-itemset.

Property 3: Given a transaction database *D*, the number of *vertices* in an AR-*graph* is no more than $|I| = n$ and that a graph with n vertices can have at most $n(n-1)/2$ edges.

Property 4: If $\{X_1, X_2, X_3, \ldots, X_k\}$ are frequent itemsets and the $sup(X_1) > sup(X_2) > sup(X_3) \ldots sup(X_k)$ then a maximal sub-graph cycle for X_1 is found before X_2.

AR-graph edges are assigned weights computed from the co-occurrence frequency counts of items from the transaction database. Edges connected through frequent items have larger weights as compared with edges connected by infrequent items. Since infrequent items do not play any role in discovering maximal frequent patterns therefore, it is represented either by no edges or smaller edge weights.

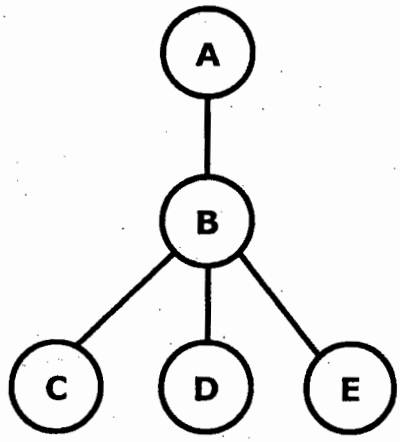
First sub-graph cycle itemset X_1 consists of all edges having maximum weight therefore they have largest support count. Similarly, the support count of items corresponding to second cycle has larger support as compared with third cycle and so forth.

4.3 All Path Source-to-Destination Tree

An AR-graph introduced in the previous section can be employed to generate MFIs. The formation of cycle in AR-graph signals the development of MFIs. It is then required to identify all cycles in the AR-graph after determination of an edge that causes a cycle. This can be carried out by designating any end point vertices of the edge that makes a cycle in the AR-graph as a source vertex and another as destination vertices. An all-path-source-to-destination-tree (ASD-tree) is constructed on first traversing all the nodes of AR-graph that can be reached from the source node where the traversal is starting and once these node shave been traversed, traversing the nodes at the next level that can be reached from those nodes and so on. This traversal stops when either a destination node is reached or a node cannot be expanded further. Given source vertex (v_s) and destination vertex (v_t) in an AR-graph, an ASD-tree is a multiple source-to-destination paths tree with v_s as its root node and v_t as leaf node such that a node may not be repeated in any path from root node to leaf node. An ASD-tree is defined as follows:

Definition 10 (all-path-source-destination tree (***ASD-tree***)) An ASD-Tree is general tree T with a finite set of one or more nodes such that there is one designated node r, called root of T, and the remaining nodes are partitioned into $n \geq 0$ disjoint subsets T_1, T_2, ..., T_n, each of which is a tree, and whose roots are children of r such that all nodes in each path (from root node to leaf nodes) are distinct.

For a maximal frequent itemset m_i, all items included in m_i can be obtained by finding all nodes along source (root) to destination paths in an ASD-tree. The source vertex is the root node of the ASD-tree. All vertices connected to the source vertex in the AR-graph are then added as child nodes to the root node of the tree. The ASD-tree is constructed using breadth-first approach. A node is only inserted as child to a parent if that node does not exist from its parent to root. A leaf node is not extended further if that node contains the destination vertex. All AR-graph cycles corresponds to source to destination paths in ASD-tree. An ASD-Tree and its data structures are shown in the Figure 7.



(a)

		Par	Path
0	A	null	
1	B	0	
2	C	1	
3	D	1	
4	E	1	Y
5			

(b)

Figure 7: The ASD-tree data structures

A simple 2-dimensional dynamic array with 3 columns can be used as the underlying data structures. The first column of the array stores the data elements, second column contains a logical pointer to the parent node, and the third column contains a Boolean value indicating a whether a path is found or not.

4.4 Mining Maximal Frequent Itemsets

Constructing an AR-graph makes certain that subsequent discovery of frequent itemsets can be performed with a compressed data structure. Different methods can be employed to derive desired maximal frequent itemsets. In this section, we develop methods to search the compact information stored in an AR-graph to discover *top-most* and *top-k* frequent itemsets without *min-sup* threshold. The procedure is explained by examination of our running example. In Sect. 4.4.1, the implementation details of finding *top-most* MFI and the concept of mining *top-k* MFIs will be discussed in Section 4.4.2.

4.4.1 Mining Top-Most MFI

The task of finding the *top-most* MFI can be defined as detecting a *maximal circuit* in the AR-graph. It is based on the observation that by selecting the edges with maximum weight produces a circuit. A maximal circuit corresponds to an MFI because edge weight between two vertices is larger only when frequent items have higher relative frequencies. By employing greedy approach a maximal circuit can be produced which correspond to the vertices of the circuit representing the *top-most* maximal frequent itemset. The definition for the maximal circuit is as follows.

Definition 11 (maximal circuit) A *maximal circuit* is a path (containing at least three vertices) from a vertex to itself in such a way that at each vertex an edge having

maximum weight is selected from an AR-graph such that there is an edge between each vertex and its successor and all the vertices and edges are distinct.

A maximal circuit is a close walk not necessarily ending at the initial vertex. This walk ends when a vertex is being visited twice. Finding maximal circuit algorithm for *top-most* maximal frequent itemset is shown as follows.

Algorithm 2: Finding *top-most* maximal frequent itemset

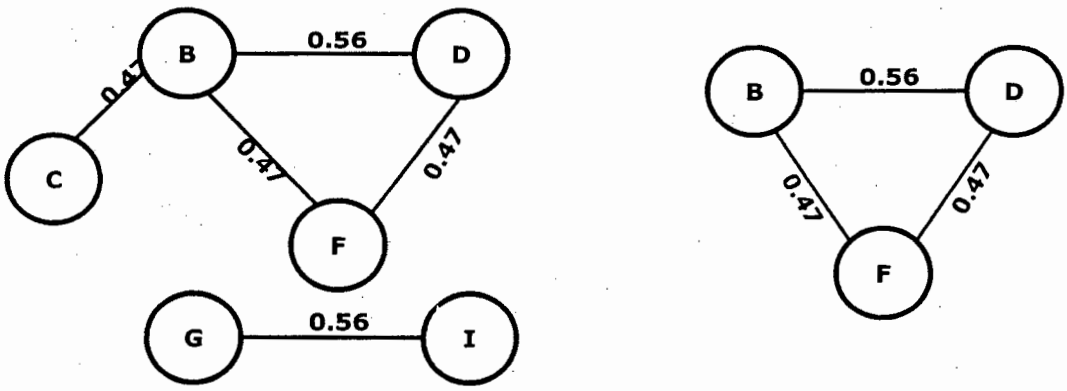
Input: *AR-Graph G = (V, E)*
Output: *Maximal Cycle PQ*

1. *Initialize graph C=(V, E) by setting V*(C) = Φ *and E*(C) = Φ
2. *Sort all edges E(G) by decreasing order of weight*
3. $e_{i,j}$ = *edge with maximum weight from E(G)*
4. $V(C) = V(C) \cup v_i \cup v_j$
5. $Degree(v_i)$++; $Degree(v_j)$++
6. $E(C)=E(C) \cup e_{i,j}$
7. ***repeat***
8. $e_{i,j}$ = *edge with maximum weight from E(G) that not yet considered*
9. $V(C) = V(C) \cup v_i \cup v_j$
10. $Degree(v_i)$++; $Degree(v_j)$++
11. $E(C)=E(C) \cup e_{i,j}$
12. ***until*** ($v_i \in V(C) \,||\, v_j \in V(C)$)
    /* *Pruning step* */
13. *Arrange all n vertices in a priority queue PQ in decreasing order of degree*
14. ***for*** ($i = 1$; $i <= n$; i++)
15. ***if*** $Degree(v_i) < 2$
16. *remove(PQ,* v_i*)*
17. *recalculate degree of remaining vertices*
18. *update(PQ)*
19. ***end***
20. ***end***
21. ***return*** *PQ* /* *PQ is left with only those vertices having degree greater than or equal to 2 representing MFI.* */

Analysis. This algorithm for finding *top-most* maximal frequent itemset utilizes the simple fact that the vertices connected by the edges having maximum weight may guide towards finding maximal frequent itemset. The algorithm starts with sorting all AR-graph edges in decreasing order of edge weight. At each step next edge that has not yet

been considered is selected and its endpoint vertices (v_i and v_j) are added to a sub-graph V(C). The loop terminates, in this phase of algorithm, when a cycle is constructed in the V(C). The cycle is detected when both the endpoint vertices of a selected edge are found present simultaneously in the V(C).

In the pruning step, vertices which are not part of maximal circuit are removed. All such vertices have degree less than two. This is implemented using a priority queue (PQ). PQ contains all vertices arranged in increasing order of vertex degree. The degree of all vertices is examined step-by-step. At each step, any vertex having degree less than two is removed from the PQ and degree of the rest of vertices is recalculated and the PQ is updated. The *for* loop terminates when the degree of all graph vertices of V(C) have been checked. The PQ is left with the vertices forming maximal cycle corresponding to items of the *top-most* maximal frequent itemset. The maximal circuit for the AR-graph of Figure 6 is shown in Figure 7. Figure 7-a shows the sub-graph before pruning step and Figure 7-b depicts the sub-graph representing the *top-most* maximal frequent itemset {B, D, F} after pruning.



(a) Maximal circuit before pruning (b) Maximal circuit after pruning

Figure 8: The illustration of *top-most* maximal frequent itemset {B, D, F}

Let's now study the complexity of these algorithms. The Algorithm 1 scans the transaction database only once and produces a condensed data structure AR-graph. Mining the *top-most* frequent pattern process is then carried out on the AR-graph. An AR-graph is typically much smaller than the original size of the database. Moreover, a maximal pattern is usually very smaller than the total number of item in database because it consists of only few frequent items. Thus, the subsequent mining process directly discovers the *top-most* maximal frequent itemset. Since the mining operations mainly consist of following edges with maximum edge weights, therefore mining is less costly as compared with the algorithms that generate and test a very large number of candidate itemsets. Thus the algorithm is efficient.

It is observed that in the case of a dense database (which contains a large number of items per transactions) the size of the AR-graph is quite small. This is because the graph size (number of vertices) depends on the number of items not on number of transactions or their length. For example, for a database of 1000 items an AR-graph will have 1000 vertices and according to the Property 5 the number of at most edges are $n(n-1)/2$.

4.4.2 Mining top-k Maximal Frequent Itemsets

The AR-graph introduced in Section 4.2 can be employed to generate *top-k* maximal frequent itemsets without *min-sup* threshold. Contrary to specifying the subtle *min-sup* this algorithm only requires the desired count (k) of MFIs. The *top-k* parameter refers to the k number of required MFIs. This approach reduces the huge set of frequent itemsets as produced by most of the current frequent pattern mining approaches. Large values of k are undesirable to work with as they may generate a huge number of MFIs. Since user

required MFIs are few in numbers therefore the value of k should be kept as small as possible.

Initially the database is scanned to produce 2-itemset association using an adjacency matrix M. The labels of matrix M represent items and the cell values represent the frequency count of two items. This frequency count is then updated to *association ratio* using the measure $M[i, j] = P(M[i, j])/(1- P(M[i, j]))$ where $P(M[i, j])=f(x_i, x_j)/|D|$. Following that all the 2-itemsets and their corresponding association ratios are arranged in decreasing order of association ratio called 2-itemset association ratio list L. An AR-graph is then constructed incrementally by drawing tuples one-by-one from L and adding as an edge to the AR-graph. The formation of a cycle is tested at each step. On detection of a cycle, one or more MFIs can then be retrieved by finding all source to destination paths in ASD-tree. A high level pseudo-code for finding *top-k* MFIs is shown below.

Algorithm 3: Finding *top-k* maximal frequent itemsets

Input: *Database* ***D****, Desired number of MFIs* ***k***
Output: *Top* ***k*** *MFIs*

1. *AR-Graph* $G = (V, E)$; $V(G) = \Phi$; $E(G) = \Phi$;
2. *List* $L[n, 3] = \Phi$ /* *first two columns of array L contain items and the third column contains association ratio* */
3. *Tree* $T[n, 3] = \Phi$ /**Initialize ASD-tree as 2-dimensional array* */
4. $M[n, n] = \Phi$
5. *Scan database* ***D*** *to generate all 2-itemsets and compute frequency count of each itemset to be stored in an adjacency matrix* ***M***
6. *Update all frequency counts using association ratio measure* $M[i, j] = P(M[i, j])/(1- P(M[i, j]))$ *where* $P(M[i, j])=f(x_i, x_j)/|D|$
7. *Sort* ***M*** *in decreasing magnitude of association ratio called 2-itemset association ratio list* ***L***
8. ***for each*** *row* $\in$ ***L***
9. $[I_1, I_2] = L.read$
10. ***if*** *(adding edge between* I_1 *and* I_2 *does not create a cycle in* ***G***)
11. *Add* I_1 *and* I_2 *as vertices to* ***G***
12. *Create edge between* I_1 *&* I_2 *and assign sequence number as edge weight*
13. ***else***

```
14.            Designate I1 as source and I2 as destination vertices
15.            Construct an all-path-source-destination tree (ASD-tree)
16.            MFIs = Print all vertices along each source-to-destination
                       path in the ASD-tree
17.            Update k
18.            if (k number of MFIs found)
19.                stop
20.            end if
21.        end if
22.    end for
```

Analysis. The algorithm described above takes time $O(n(\frac{m!}{2!(m-2)!})) = O(n m^2)$ to scan the database D and to produce the adjacency matrix M where n is the number of transactions in the database and m is the transaction length. The frequency counts calculated in M can be converted to association ratios in time $O(n^2)$. We next sort M in time $O(n^2)$ to produce 2-itemset transaction ratio list L. The AR-graph is constructed in time $O(n(n-1)/2) = O(n^2)$ where n is number vertices in the graph. Given an AR-graph G, a source vertex v_s and a destination vertex v_t we can construct an all-path-source-destination tree in time $O(n^2)$. Then we can compute paths from the root node to leaf nodes in the ASD-tree representing cycles in the graph G in time $O(n)$. Total running time for the algorithm is $O(nm^2 + n^2 + n^2 + n^2 + n^2 + n) = O(nm^2 + 4n^2 + n)$.

a) A Running Example to Generate top-k MFIs

The step-by-step process of generation of top five ($k = 5$) MFIs for the sample transaction database of Table 1 using Algorithm 3 is illustrated below. The algorithm accepts the database and a value for parameter k as input. It then generates all frequent 2-itemsets and computes their association ratios. The 2-itemsets list is then sorted in non-increasing order of association ratio magnitude as shown in the Table 8.

Table 8: Two-itemsets association list using example database of Table 1

#	Itemsets	AR	#	Itemsets	AR
1	BD	0.563	27	GJ	0.087
2	GI	0.563	28	IJ	0.087
3	BC	0.471	29	AC	0.042
4	BF	0.471	30	AJ	0.042
5	DF	0.471	31	AP	0.042
6	EG	0.389	32	BJ	0.042
7	EI	0.389	33	BP	0.042
8	BI	0.316	34	DP	0.042
9	DI	0.316	35	EK	0.042
10	CD	0.250	36	FG	0.042
11	CE	0.250	37	FJ	0.042
12	CI	0.250	38	FP	0.042
13	AB	0.190	39	GH	0.042
14	BE	0.190	40	GK	0.042
15	BG	0.190	41	HI	0.042
16	CF	0.190	42	HJ	0.042
17	CG	0.190	43	HO	0.042
18	DE	0.190	44	IK	0.042
19	DG	0.190	45	JK	0.042
20	AD	0.136	46	JP	0.042
21	AF	0.136	47	KL	0.042
22	JO	0.136	48	KM	0.042
23	DJ	0.087	49	LM	0.042
24	EF	0.087	50	MN	0.042
25	EJ	0.087	51	MO	0.042
26	FI	0.087	52	NO	0.042

Following that an AR-graph is constructed incrementally by drawing each tuple from 2-itemset association ratio list in each iteration. The first tuple consists of items B and D are drawn from the list and added to the AR-graph as shown in Figure 9-a.

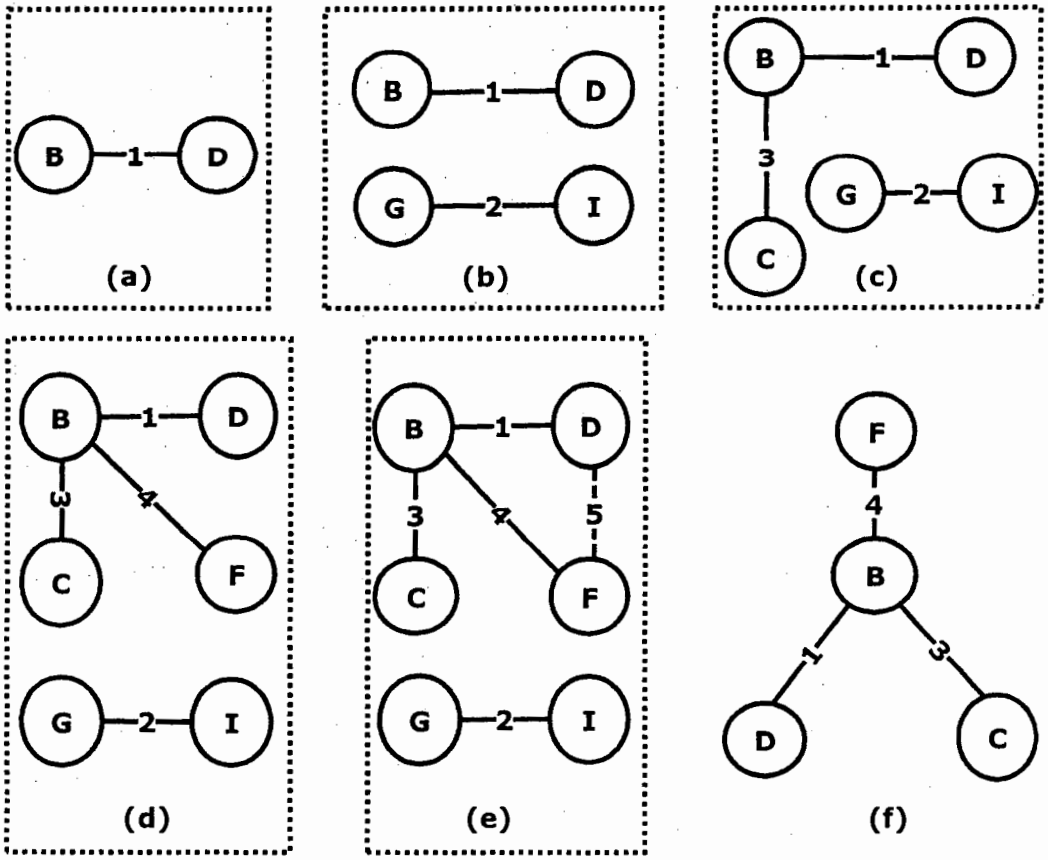


Figure 9: Step-1 of construction of AR-graph and ASD-tree

The items B and D are made the graph vertices and an undirected edge is created between the two vertices. The sequence number of the list is assigned as a weight to each edge. The process of addition of first five edges added to the AR-graph is shown in the Figure 9-a, 9-b, 9-c and 9-d. Before addition of each edge to the AR-graph it is checked whether the edge under consideration creates a cycle in the graph or not. The vertices and edges are added to the graph if the edge to be added does not create a cycle. If an edge is involved in creation of cycle(s) then an ASD-tree is constructed to find all paths corresponding to the AR-graph cycles before the addition of that edge to the AR-graph. The addition of edge 5 creates a cycle in AR-graph as shown in Figure 9-e. An ASD-tree

(Figure 9-f) is constructed by considering F as a source vertex and D as a destination vertex. The first MFI {F, B, D} corresponds to the source to destination path in ASD-tree as shown in Figure 9-f. Having constructed ASD-tree the edge 5 is now added to the AR-graph and algorithm moves on to draw another tuple from the 2-itemset association ratio list.

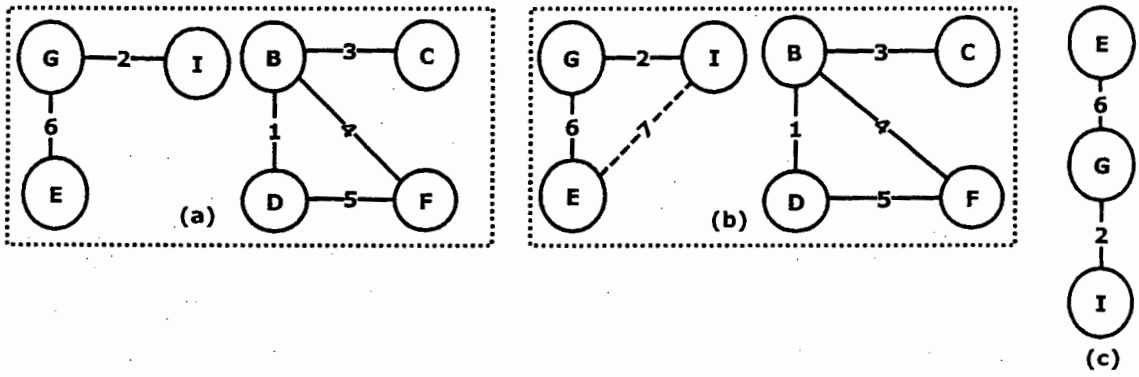


Figure 10: Step- 2 of construction process of AR-graph and ASD-tree

As can be seen in Figure 10-a, the addition of edge 6 is not going to create any cycle in the AR-graph. The algorithm moves on to fetch next tuple from the list and checks for the formation of a cycle in the graph. This edge can create another cycle in the graph (Figure 10-b). Before addition of this edge to the graph an ASD-tree is constructed as shown in Figure 10-c. This gives the second MFI {E, G, I} corresponding to the tree nodes for the source to destination path.

Similarly, the addition of edge 8 does not create any new cycle in the graph (Figure 11-a). Next, edge number 9 is drawn that can create a new cycle in the AR-graph as shown in Figure 11-b. The algorithm draws the ASD-tree considering I as a source vertex and D as destination vertex. The resultant ASD-tree is shown in the Figure 11-c. Two source to

destination paths {I, B, D} and {I, B, F, D} can be seen in Figure 11-c corresponding to third and fourth MFIs respectively.

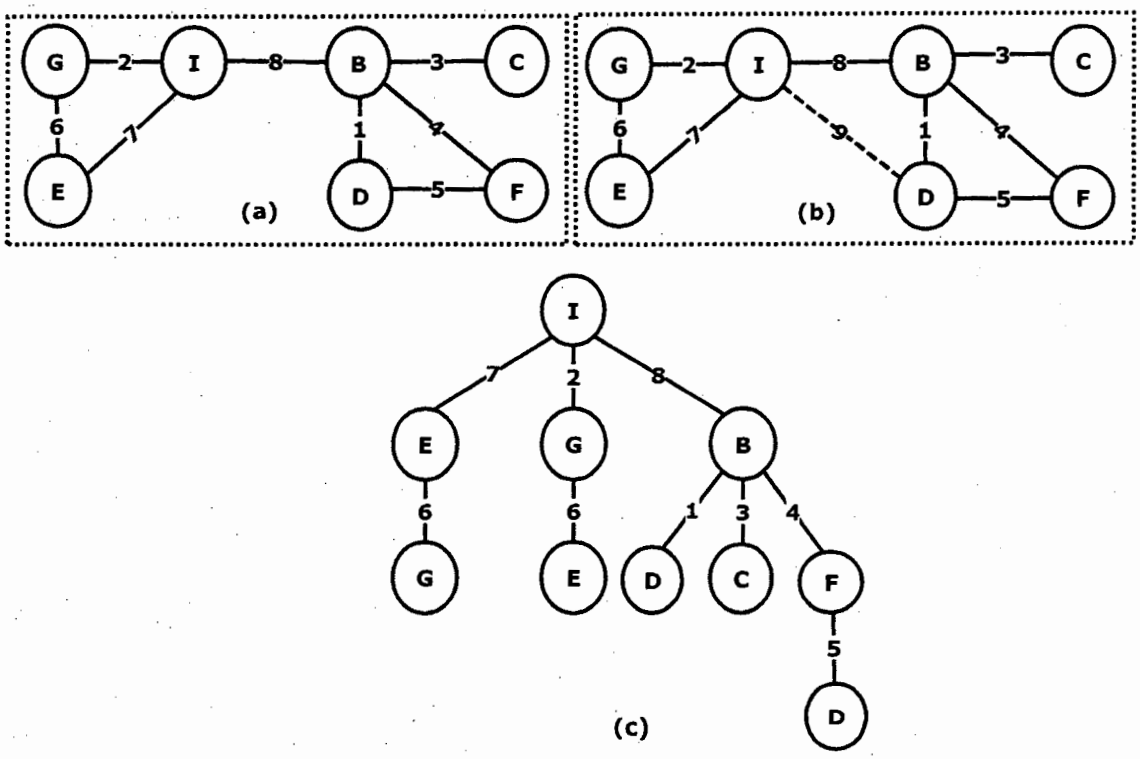


Figure 11: Step- 3 of construction process of AR-graph and ASAD-tree

Following this next tuple is drawn from the list and checked for the formation of potential cycle in AR-graph as shown in the Figure 12-a. The addition of edge CD, number 10, can create another cycle in the graph. Thus, making D as source vertex and C as destination vertex, the AR-graph is traversed to construct ASD-tree. The corresponding ASD-tree is shown in the Figure 12-b with three source to destination paths.

The {D, B, C}, {D, I, B, C}, and {D, F, B, C} are the corresponding MFIs for the three source to destination paths of the ASD-tree in Figure 12-b. Since the required number (k=5) MFIs are found therefore the algorithm stops.

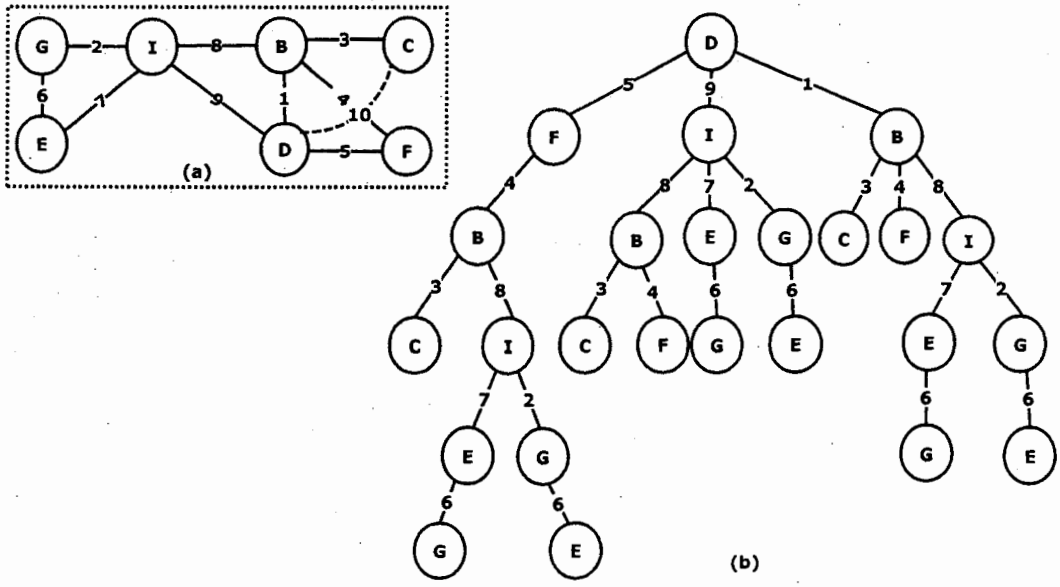


Figure 12: Step-4 of construction of AR-graph and ASD-tree

After computing the sum of weights of each ASD-tree edges for the corresponding MFI paths, the MFIs are arranged in the order of importance. The discovered MFIs are shown in the Table 9.

Table 9: Top-*k* MFIs result

S. No.	Top-*k* MFIs	Weight
1	{F, B, D}	10
2	{D, B, C}	14
3	{E, G, I}	15
4	{I, B, D}	18
5	{D, F, B, C}	22
6	{I, B, F, D}	26
7	{D, I, B, C}	30

The Algorithm 3 produces the solution without user provided *min-sup* threshold. This shows the good performance and accurate approach of the Algorithm 3. This is very practical and useful approach involving limited computational efforts, when it is required to mine top few maximal frequent patterns.

4.5 Experimental Study

The effectiveness of the proposed methods using an example was illustrated in the previous section. This section discusses the results of empirical study performed on different datasets to assess the efficiency of our approach. The simulation environment of our experimental evaluation and the selected datasets are described in Section 4.5.1 while Section 4.5.2 describes the organization of data in the standard datasets. The experiments performed on the transaction clustering are presented in Section 4.5.3. In Section 4.5.4 and Section 4.5.5 conduct empirical studies on the selected datasets to discover top-most and top-k MFIs and assess the effectiveness and performance of proposed algorithms.

4.5.1 Datasets

Both real and synthetic datasets were selected to test the performance of the proposed algorithms. There is a wide use of these datasets in mining frequent patterns studies. All experiments were carried out on a 3.2 GHz IBM compatible PC with 1 GB of RAM, running Windows XP to evaluate the performance of the proposed methods. Some characteristics of the selected datasets are shown in Table 10.

Table 10: Parameters of synthetic datasets

S. No.	Dataset	Number of Items	Number of Transactions	Average Transaction Size
1	T10I4D100K	1000	100,000	10
2	T40I10D100K	1000	100,000	40
3	Mushroom	120	8,124	23
4	Connect	130	67,557	43
5	Retail	16470	88,162	10

The first dataset T10I4D100K contains 1000 items and 100k total number of transactions. The average transaction length and average potentially maximal frequent itemsets length of this dataset are 10 and 3, respectively. The dataset is sparse (average transaction length is shhort) and the frequent itemsets are also short and not enormous. The second synthetic dataset we used is T40I10D100K having 1000 items and 100k total number of transactions. The length of average transaction and average potentially maximal frequent itemset are 40 and 3 respectively. This dataset includes large number of frequent itemsets when the support threshold is set to a small value. This is a dense dataset containing plentiful mixture of short and long frequent itemsets. The third synthetic dataset used is Mushroom with 120 items and 8124 transactions. The average transaction size is 23. This is a sparse dataset containing few short frequent itemsets. The fourth dataset is called Connect constructed over 130 items. The dataset has the longest average transaction size i.e., 43 which categorize it as a dense dataset. The fifth and final dataset is Retail with largest number of items i.e., 16470. The average transaction length is 10 which make it sparse.

4.5.2 Database Representation

To compute the frequent itemsets it is required to access the database. The database representation is one of the most important considerations in implementation of majority of algorithms. This factor directly affects the performance of generating and counting frequent itemsets. Generally transaction databases can be organized in *horizontal* or *vertical* representations. Most algorithms prefer *horizontal* layout, in which the database is organized as a set of rows (transactions) and each row having a set of items. The order of transactions in the database and items in transaction is immaterial. On the other hand in *vertical* layout each row represents an item and a set of transaction identifiers (tids) are used to associate transactions with each item.

Apart from this initial database representation many algorithms use different assumptions and transformations. For example, Mafia assumes that each transaction of the database should be in a total lexicographic order of the items. If item i occurs before item j in the ordering i.e., $i \leq j$, this ordering is used to enumerate the item subset lattice.

The DLG* requires *transformation* of underlying database into a special representation called bit vectors. Each database transaction is scanned and a bit vector is built for each database item. The length of a bit vector is the number of transactions in the database. These assumptions and transformations are vulnerable and time consuming. A user unaware of basic assumptions will not be able to generate correct results. Similarly, transformation of database to another format consumes time and resources. Moreover, a relational database defines a relation as two-dimensional table of data, consisting of attributes and tuples. The order of attributes and tuples is be irrelevant in a relation.

Thus, imposing any restrictions before mining, such as assuming data in lexicographic order, is violation of basic database principles.

The algorithm number 2 and 3 proposed in this study for mining frequent itemsets do not impose any assumptions or require any kind of special transformation of underlying data. The algorithm can be applied directly to the horizontal database layout that makes the proposed approach more efficient by saving transformation efforts, resources, and time.

4.5.3 Transaction Clustering

The basic intuition of mining frequent patterns without minimum support threshold can best be explained using concept of transaction clustering. Items that are common to many transactions, termed as frequent itemset, can form different clusters of items. A cluster is a set of items that appear near to each other due to a proximity measure in such a way that an item in a cluster is closer to one or more other items in the cluster than to any item not in the cluster. Strong associations among the items bring them closer to one another forming a cluster separate from other items. Thus, transactions sharing same items have greater chance of belonging to the same cluster and a top-down approach can reasonably be used for mining frequent itemsets without minimum support.

The majority of candidate generation methods are based upon bottom-up approach, i.e., if any length k itemsets found frequent in the database, it's all length $(k + 1)$ supersets are generated and tested until no further candidate frequent itemsets can be generated. A transaction cluster shows that it is sufficient to generate only length 2-itemsets for retrieving MFIs and there is no need to generate length $(3, 4, \ldots, n)$-itemsets.

Association ratio is very useful to represent transaction clusters. This can be shown by drawing an XY-scatter plot between database items and their association ratios. Single

database scan is enough to construct all length 2-itemsets and their corresponding association ratios. Different maximal frequent itemsets will appear as separate clusters on the scatter plot area. This is due to association ratio measure that brings highly correlated items close to each other and moves infrequent itemsets away from frequent itemsets.

Figure 13 shows the XY-scatter plots for the datasets in Table 10. To avoid the clutter all plots are drawn for the top 50 pair of items after ranking all itemsets in decreasing order of association ratio magnitude.

In each scatter plot we can observe some clusters of items distinctively. Some clusters having few items are located in high association ratio region and a cluster having large number of items is located in the low association region. This indicates that frequent itemsets are few in number and a large number of infrequent itemsets. The overlapping of items also increases towards the low association ratio regions. This is because infrequent items are in large number having low association ratios. On the other hand, clusters representing top-k MFIs are few and are located distinctly far apart from infrequent itemsets. This shows the practicability of designing of top-down approach for mining frequent itemsets. This approach can discover MFIs without using *min-sup* threshold directly and easily.

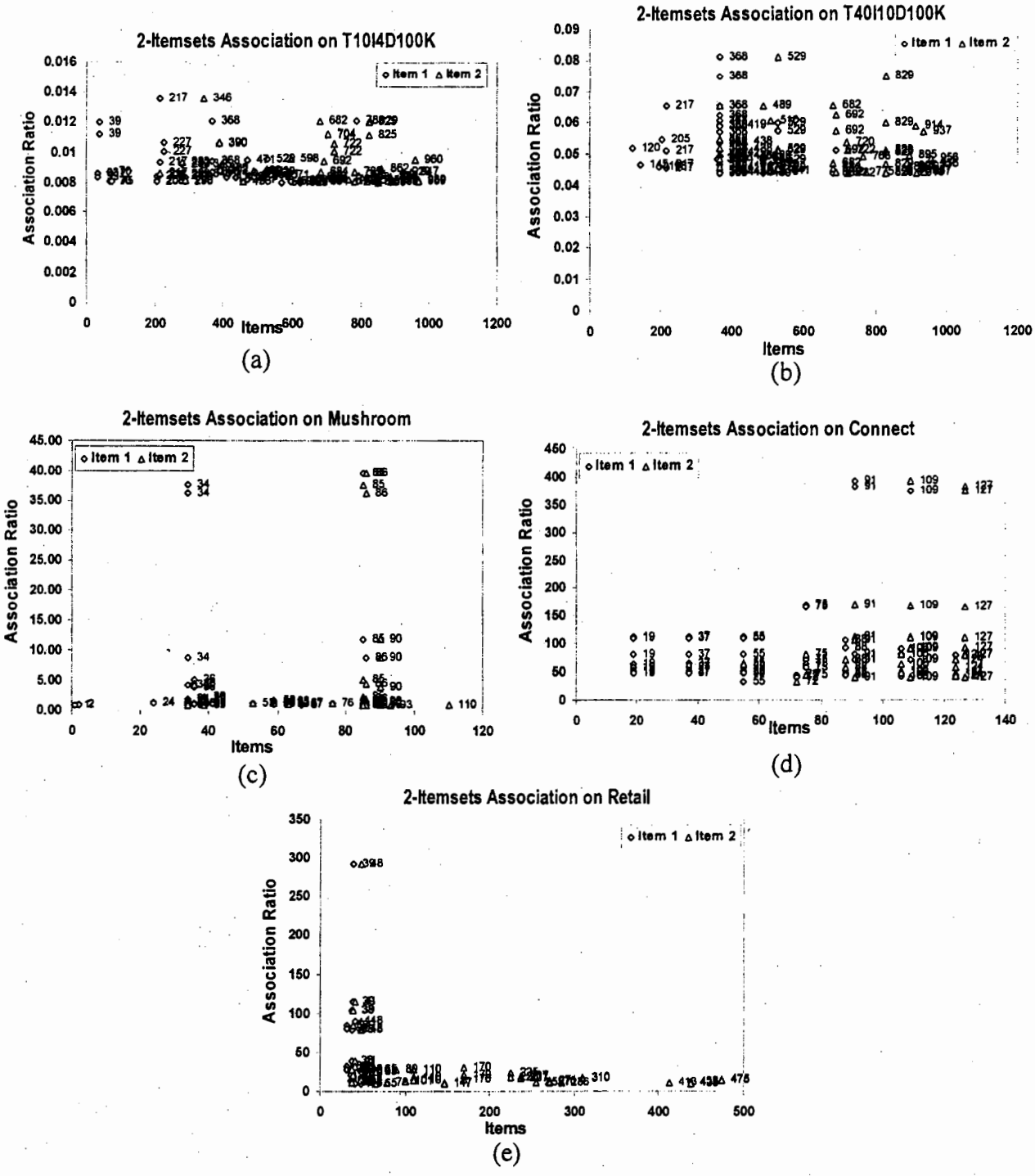


Figure 13: XY-scatter plots drawn on various datasets

As can be seen in Figure 13-d, an XY-scatter plot drawn for the Connect dataset, that an itemset {91, 109, 127} having highest association ratio forms a separate cluster from the rest of the data points. Similarly, another itemset {75, 91, 109, 127} can also be seen as the second cluster. This is due to the association ratio computed in such a way that it brings highly correlated items close to each other and lays infrequent items far apart. The rest of XY-plots drawn on the T10I4D100k, T40I10D100k, Mushroom and Retail datasets show items forming clusters of frequent itemsets such as {39, 704, 825}, {368, 529, 829}, {34, 85, 86}, and {75, 91, 109, 127} in Figure 13-a, 13-b, 13-c and 13-e respectively. Figures 13-a and 13-b also indicate that these datasets have a vast majority of length two frequent itemsets. This clearly indicates that a top-down approach is very likely for mining top-most and top-k MFIs without user provided *min-sup*.

4.5.4 Experiments on Mining Top-Most MFI

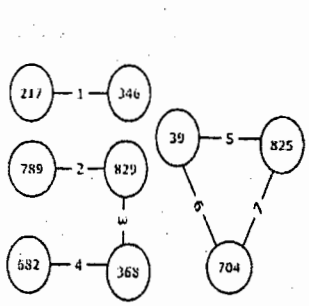
Many algorithms exist for mining MFI and are available for performance comparison. Apriori (Agrawal and Srikant, 1994) is the first and the most influential algorithm for frequent pattern mining and many of the proposed algorithms are its variants. DepthProject (Agarwal, Aggarwal and Prasad, 2000), Mafia (Bayardo, 1998) and GenMax (Gouda and Zaki, 2005) are the other high-tech algorithms for mining MFI. Tianming et al., (2007) observe that "DepthProject, Mafia and GenMax share a lot in common: search the item subset lattice in a depth-first way, apply look-a-head pruning and dynamic reordering to reduce the search space and using compression methods for fast support counting". Therefore, Apriori and MAFIA are chosen as the representatives of the frequent pattern mining algorithms. Borgelt's (2009) implementation of Apriori is used that uses a prefix tree to organize the counters for the itemsets.

To illustrate the effectiveness of our approach, we employ Algorithm 2 (TMMFI) presented in Section 4.2 along with Apriori and Mafia to discover *top-most* MFI. All three algorithms are tested on the standard datasets of Table 8. Apriori and Mafia require considerable time to tune *min-sup*, however, we consider only their CPU execution time for performance comparisons. As can be seen in Table 11, Apriori and Mafia are tested using different *min-sup* values ranging from 1% to 99.95% for finding top-most MFI. The resultant itemsets produced by Apriori and Mafia contain a huge number of patterns. As shown in Table 11, we obtain 65236 and 21692 MFIs discovered by Apriori and Mafia respectively for T40 dataset. This is because if a pattern is frequent, each of its subpattern is frequent as well and finding top MFI from such a huge result set is very difficult.

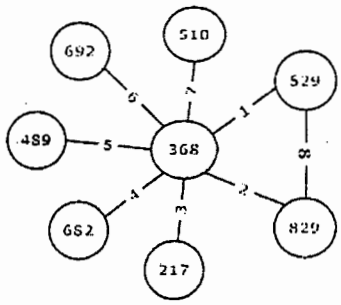
Table 11: MFIs generated by different algorithms

Datasets	Apriori				Mafia				TMMFI (Algo-2)	
	Min. Supp. (%)	# MFI	Top-Most MFI	Time (Sec)	Min. Supp. (%)	# MFI	Top-Most MFI	Time (Sec)	Top-Most MFI	Time (Sec)
T10I4D100K	1	385	39, 704, 825	2.61	1	370	39, 704, 825	18	39, 704, 825	2.5
	2	155	368	2.24	2	155	368	12		
	10	0			10	0				
	20	0			20	0				
T40I10D100K	1	65236	368, 529, 829	26.87	1	21692	368, 529, 829	98	368, 529, 829	16.65
	2	2293	368, 529, 829	11.84	2	2015	682 489 368	59		
	10	82	368	8.01	10	82	368	43		
	20	5	368	6.98	20	5	368			
Mushroom	20	1197	34, 85, 86	1.67	20	158	24, 34, 39 85, 86, 90	2	34, 85, 86	1.63
	45	83	34, 85, 86	0.4	45	18	24, 34, 85, 86, 90	1		
	70	12	34, 85, 86	0.33	70	1	34, 36, 85, 86, 90	2		
	95	4	34, 85, 86	0.36	95	1	34, 85, 86	2		
Connect	98	21	55, 75, 127, 109, 91	4.79	98	21	55 75 127 109 91	23	91, 109, 127	4.17
	99	11	75, 91, 109, 127	4.36	99	11	75 127 109 91	23		
	99.94	4	91, 109, 127	4.71	99.40	4	127 109 91	23		
	99.95	2	91, 109, 127	4.42	99.50	2	127 109 91	22		
Retail	1	159	39, 41, 48	2.53	1	78	39, 48, 89	11	39, 41, 48	89
	2	55	39, 41, 48	2.36	2	20	32, 39, 48	10		
	10	9	39	2.36	10	5	39, 48	11		
	20	3	39	2.29	20	1	39, 48	11		

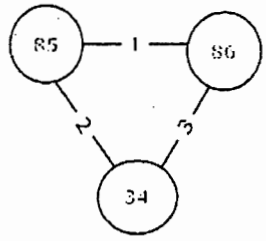
We now investigate the performance of proposed algorithms against Apriori and MAFIA on all five standard datasets. Table 11 displays the results produced by the respective algorithms. As we know that our algorithm does not require any input parameter whereas the rest of methods require setting of appropriate *min-sup* thresholds. The *top-most* maximal frequent itemsets discovered by Algorithm 2 are shown in the last two columns of the Table 11. Similar results could be obtained from rest of the algorithms but they require considerable time and efforts to tune *min-sup* over a wide range. Further, in rest of the algorithms we do not have any sway over the number of resultant itemsets. Thus users have to manually filter out desired results. Our approach liberates users of this tedious job of tuning *min-sup* and gives control over the number of results produced.



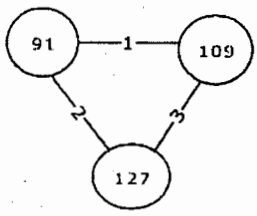
(a) Top-most cycle (7-5-6) on T10



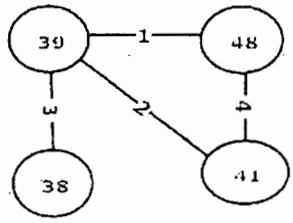
(b) Top-most cycle (8-2-1) on T40



(c) Top-most cycle (3-2-1) on Mushroom



(d) Top-most cycle (3-2-1) on Connect



(e) Top-most cycle (4-2-1) on Retail

Figure 14: AR-graphs and maximal cycle sub-graphs of standard datasets

The association ratio graphs and maximal graph cycles generated by the Algorithm 2 using standard datasets are shown in Figure 14. As can be seen, after pruning unnecessary vertices we can obtain maximal graph cycles from AR-graphs, representing the required *top-most* maximal frequent itemsets. In case of Mushroom (Fig. 14-c) and Connect (Fig. 14-d) there is no need of pruning.

We now examine the performance of the Algorithm-2, as compared with Apriori and Mafia on datasets specified in Table 10. In Figure 15, we present the comparative performance of the proposed algorithms on five standard datasets. Time to tune different support thresholds for Apriori and Mafia are not included in the total time. It is apparent to see that Algorithm-2 outperforms Mafia when dataset is sparse. The reason is that Algorithm-2 generates only 2-itemsets and their associations and does not generate length 2, 3, …, *n* itemsets. Much execution time is saved by avoiding calculating and maintaining itemsets support counts. Our algorithms construct and employ an undirected

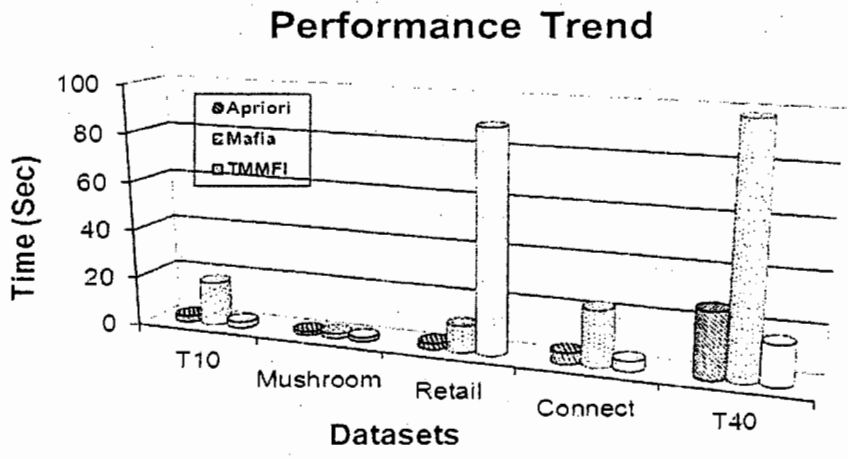
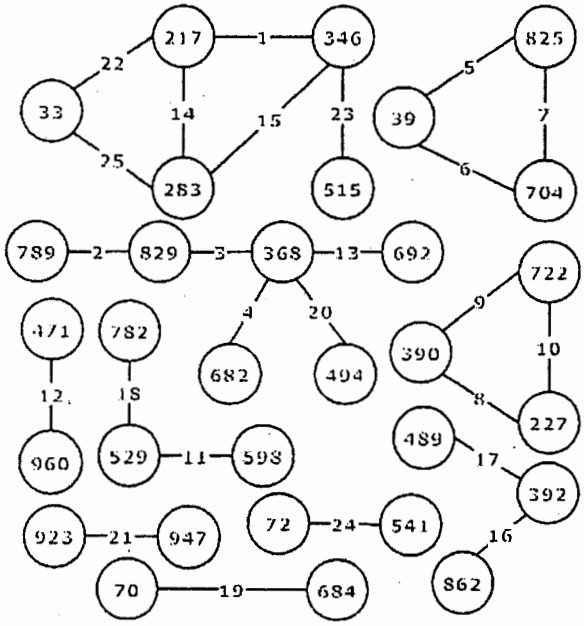


Figure 15: Time of execution on different datasets

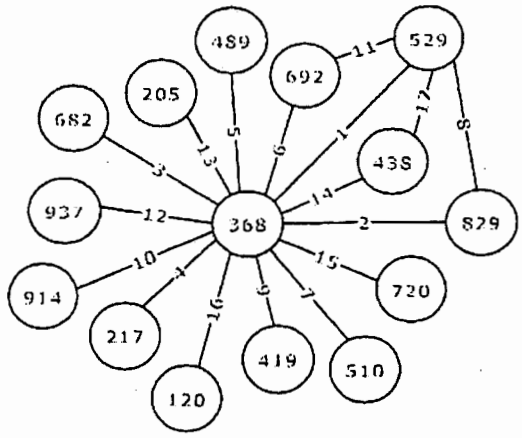
weighted graph and discover maximal frequent itemsets directly using AR-graph. This greatly reduces the memory requirements as sparse matrix can easily be used to represent the undirected graph reducing the problem of matrix sparsity.

4.5.5 Experiments on Mining Top-k MFIs

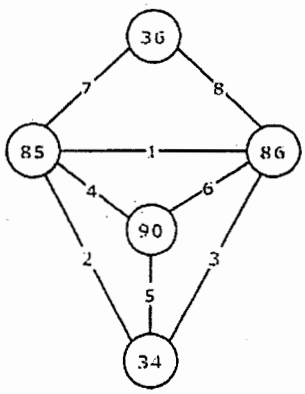
We now perform Algorithm 3 (TKMFI) on the standard datasets of Table 10 to retrieve *top-k* MFIs. The value of *k* is set to 5 knowing that maximal itemsets are very few. The AR-graphs generated by the Algorithm 3 for each dataset are shown in Figure 16. All



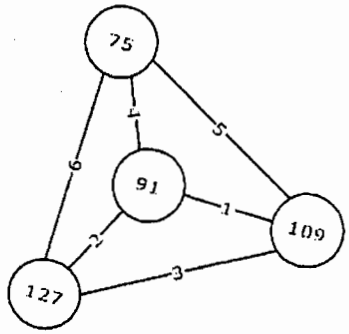
(a) Top-5 cycles (7-5-6), (10-8-9), (15-14-1), (25-14-22), (25-15-1-22) on T10



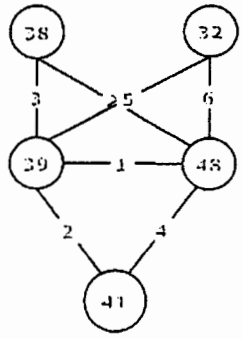
(b) Top-5 cycles (8-1-2), (11-1-6), (11-6-2-8), (17-1-14), (17-8-2-14) on T40



(c) Top-5 cycles (3-2-1), (5-4-2), (5-3-1-4), (6-1-4), (6-3-5) on Mushroom



(d) Top-5 cycles (3-2-1), (5-4-1), (5-4-2-3), (6-4-2), (6-5-3) on Connect



(e) Top-5 cycles (4-2-1), (6-5-1), (6-4-2-5), (7-3-1), (7-3-2-4) on Retail

Figure 16: AR-graphs showing top-5 graph cycles

edges in each graph are assigned unique numbers in sequence denoting the order in which they are added to the AR-graphs. On addition of every edge to the AR-graph, the algorithm checks for the formation of a cycle involving the most recently added edge. On detection of cycle in AR-graph an ASD-tree is constructed to produce the corresponding MFI(s).

For illustration, top five MFIs discovered by constructing ASD-trees corresponding to each AR-graph cycle are shown in Figure 16. The first cycle is formed after the addition of edge number 7, 8, 3, 3, and 4 for the datasets T10, T40, mushroom, connect and retail. The MFIs corresponding to the first cycle are {39, 704, 825}, {368, 529, 829}, {34, 85, 86}, {91, 109, 127} and {39, 41, 48} as shown in Figure 16-a, 16-b, 16-c, 16-d, and 16-e, respectively. Similarly, top-five maximal frequent itemset results after the formation of subsequent cycles are summarized in the Table 12. The left most column shows the edge number after which the cycle is created and the rest of columns show top-5 MFIs found after addition of specified edge number for each dataset. From the table, it can be seen that the top few MFIs can be discovered by traversing the cycle using ASD-tree.

Table 12: Results of top-5 MFIs discovered from different datasets

Cycle #	T10I4D100K		T40I10D100K		Mushroom		Connect		Retail	
	After addition of edge #	MFIs	After addition of edge #	MFIs	After addition of edge #	MFIs	After addition of edge #	MFIs	After addition of edge #	MFIs
1	7	39, 704, 825	8	368, 529, 829	3	34, 85, 86	3	91, 109, 127	4	39, 41, 48
2	10	227, 390, 722	11	368, 529, 692	5	34, 85, 90	5	75, 91, 109	6	32, 39, 48
3	15	217, 283, 346	11	368, 692, 529, 829	5	34, 85, 86, 90	5	75, 91, 109, 127	6	32, 39, 41, 48
4	25	33, 217, 283	17	368, 438, 529	6	85, 86, 90	6	75, 91, 127	7	38, 39, 48
5	25	33, 217, 346, 283	17	368, 438, 529, 829	6	34, 86, 90	6	75, 109, 127	7	38, 39, 32, 48

The technique that we have proposed in this work has the competitive efficiency in narrowing the search to only *top-most* and *top-k* maximal frequent itemsets by avoiding

superset generation and checking altogether. Mining without *min-sup* threshold being the prominent advantage of these algorithms makes them practical for mining top-k MFIs.

4.5.6 Mining MLFIs

Mining a subset of frequent itemsets such as maximum length frequent itemsets (MLFI) is often necessary in various real world applications. For example, a superstore may be interested to introduce a new sales policy to increase the number of items bought together by enough number of consumers, or an insurance company may want to design a package to maximize the number of insured objects to attract sufficient number of customers. Finding MLFIs can be very efficient and effective in solving such type of problems because the number of MFLFs is very small.

In the previous sections we have discovered the *top-k* MFIs. It is easy to find MLFIs by slightly modifying Algorithm-3 by comparing and eliminating the smaller length itemsets. This can be done by computing the length of itemsets. Having computed the length of itemset smaller length itemset can be removed from the result list and retaining the maximum length itemsets.

4.6 Summary

In this chapter, an important data mining problem, that is mining frequent patterns without *min-sup* threshold has been studied. To achieve this, novel data structures called AR-graph and ASD-tree are proposed. A retrieval algorithm based on breadth-first search approach is devised to extract *top-most* and *top-k* maximal frequent itemset. A novel search approach is devised to efficiently and effectively retrieve frequent patterns without specifying the *min-sup* value. This approach can both achieve high efficiency and performance as demonstrated in the empirical study on real and synthetic datasets.

Its major advantage to be a practical algorithm for mining frequent itemsets is also discussed. The methods proposed in this work are extremely powerful and practical in narrowing the search for true data mining, by relaxing the condition of the choice of minimum support threshold.

CHAPTER 5

SEMANTIC IMAGE RETRIEVAL WITH FREQUENT ITEMSET FRAMEWORK

In Chapter 4, a novel *min-sup* free method for mining *top-most* and *top-k* frequent patterns was developed. Its efficiency and accuracy as compared with other algorithms is also demonstrated. In this chapter, the application of the frequent itemset (FI) based mining approach will be studied for retrieving semantically relevant images from large image repositories.

Section 5.1, presents the disparity between the existing technologies and user requirements with respect to semantic image retrieval. Then Section 5.2, discuss the proposed system architecture. In Section 5.3, the correspondence between image retrieval system and FI is given along with its implementation details. In Section 5.4, experimental results are reported and the chapter concludes with summary in Section 5.5.

5.1 Semantic Gap

User, either professional or novice are required to organize, browse and search widespread collections of very large repositories of images. Many organizations handle large image databases for their internal use, like newspapers and magazines publishers (news, showbiz and sports images), hospitals (medical imaging), architecture & engineering designers (technical drawings), meteorological departments (weather

forecast, GIS) etc. When images are received, archivists label them with suitable captions that are likely to be useful means for retrieving the images. Keywords based searching techniques can then be used to search the image collection to retrieve semantically related images. Due to the rapid growth in personal and professional image libraries, the need of mining image databases is becoming increasingly important to enable semantic image retrieval. Semantic image retrieval has appeared as an important area in multimedia computing to enable users to retrieve relevant images from image repositories in an effective and efficient manner.

In digital image processing systems images are represented by sets of low-level image features, such as color, texture, and shape. These low-level visual features can be extracted by some basic image processing techniques automatically. Searching is then performed by applying a similarity measuring algorithm using the extracted visual features, which ranks the selected images according to a measure of similarity between the query and the target images. A wide range of machine learning and computer vision techniques have been applied to search image repositories. Typically, images are compared on the basis of features extracted from the entire image (global feature matching) or from image regions (local feature matching). Although computational techniques extract low-level features from pixel information automatically and very efficiently, however, from human reasoning point of view semantic gap still exists between the low-level image features and high-level semantic concepts. One result of the semantic gap is that the user's needs are seldom captured by the existing methods, and may be part of the reason that really practical semantic image retrieval systems (SIRS)

are yet to become popular. Thus, huge gap exists between user requirements and what technology provides. Major problems of existing approaches are;

- Rege, Dong and Fotouhi (2007) have observed that "current state-of-the-art multimedia technology lags far behind the human's ability to assimilate information at a semantic level. Consequently, most of those image retrieval systems give poor performance for semantic queries";
- humans recognize images based on high-level concepts. That is, users are familiar with natural language-like queries, such as city landscape, beach, wild life etc. and typically query multimedia database by semantics;
- that users search image database both by what they depict (objects visible in the picture, like buildings, cars, people, road) and by what they are about (high level semantic concepts visible in the picture, like city landscape);
- generally in machine learning supervised and unsupervised techniques are used to construct learning machines. Often these learning machines are specific to an object which identify and retrieve only that object;
- majority of content based image retrieval (CBIR) systems generate "low-level image features such as color, texture, shape, and motion for image indexing and retrieval. This is partly because low-level features can be computed automatically and efficiently. The semantics of the images, which users are mostly interested in, however, is seldom captured by these low-level image features. On the other hand, there is no effective method yet to automatically generate good semantic features of an image" (Zhang, Zhang, and Khanzode, 2004);

- most of semantic image retrieval and object recognition approaches work with datasets that only contain images of a certain type (Quelhas et al., 2005; Sivic, Russell, Efros, Zisserman, and Freeman, 2005; Sudderth, Torralba, Freeman, and Willsky, 2005). The images are taken by professional photographers focusing on the object of interest. These methods fail to recognize objects when provided with images containing more than one identifiable objects or occluded objects.
- an interesting and effective approach is to have image database annotated in more detail although it is more labor intensive to produce such databases. It is useful to work with images having detailed annotation of objects because current segmentation methods are not capable enough of ascertaining the edges of many objects, which are often small in size and vague especially in natural scenes.
- The use of external background knowledge can be very interesting when the context of interpretation of the involved concepts is precisely known. WordNet (Miller et al., 1990) is often used as background knowledge resource. However, the drawback is that it is difficult to get relevant information if the meaning of the searched terms is not known. The difficulty is encountered in particular in the misinterpretation problem coming from the different senses of a term.

The approach based on mining MFIs without *min-sup* proposed in the study by Salam and Khayal (2011) may also work well for semantic image retrieval system (SIRS) to resolve these problems. The proposed retrieval system is based on the frequent itemset (FI) mining paradigm. Establishing association among the image objects based on their co-occurrence and their corresponding labels can shed more light on semantic understanding of an image. Once we have a database of annotated images, computing the

association ratio between different objects and labels is useful for semantic image retrieval system.

In designing the SIRS the following objectives are considered:

1. Efficiently retrieve similar images from the image database using FI framework rather than searching the entire database.
2. Apply image retrieval in finer details without requiring learning machines, which are difficult to design and implement.
3. Rank output so as the most similar image to the query image is ranked higher.

5.2 The System Architecture

The task of SIRS is to derive similar images from the image database which are semantically analogous to a given query image. The strategy which we have adopted is shown diagrammatically in Figure 17. This method turns the semantic retrieval problem into mining frequent itemset problem, by providing an intuitive framework to search in an annotated image database. The proposed FI based method enables the application of AR-graph and ASD-tree approach (Salam & Khayal, 2011) to semantic image retrieval problem.

Image segmentation is the first step and also one of the most critical tasks of image analysis. Saux and Amato (2004) argue that division of images into variable size segments provides more semantic information than the usual global image features or dividing an image into fixed sized blocks. The ultimate objective of segmentation is to extract object feature information for further processing. Our proposed framework uses, LabelMe (Russell, Torralba, Murphy, and Freeman, 2008), an interactive publicly annotated large-scale collection of high-level online image database that provides sharing

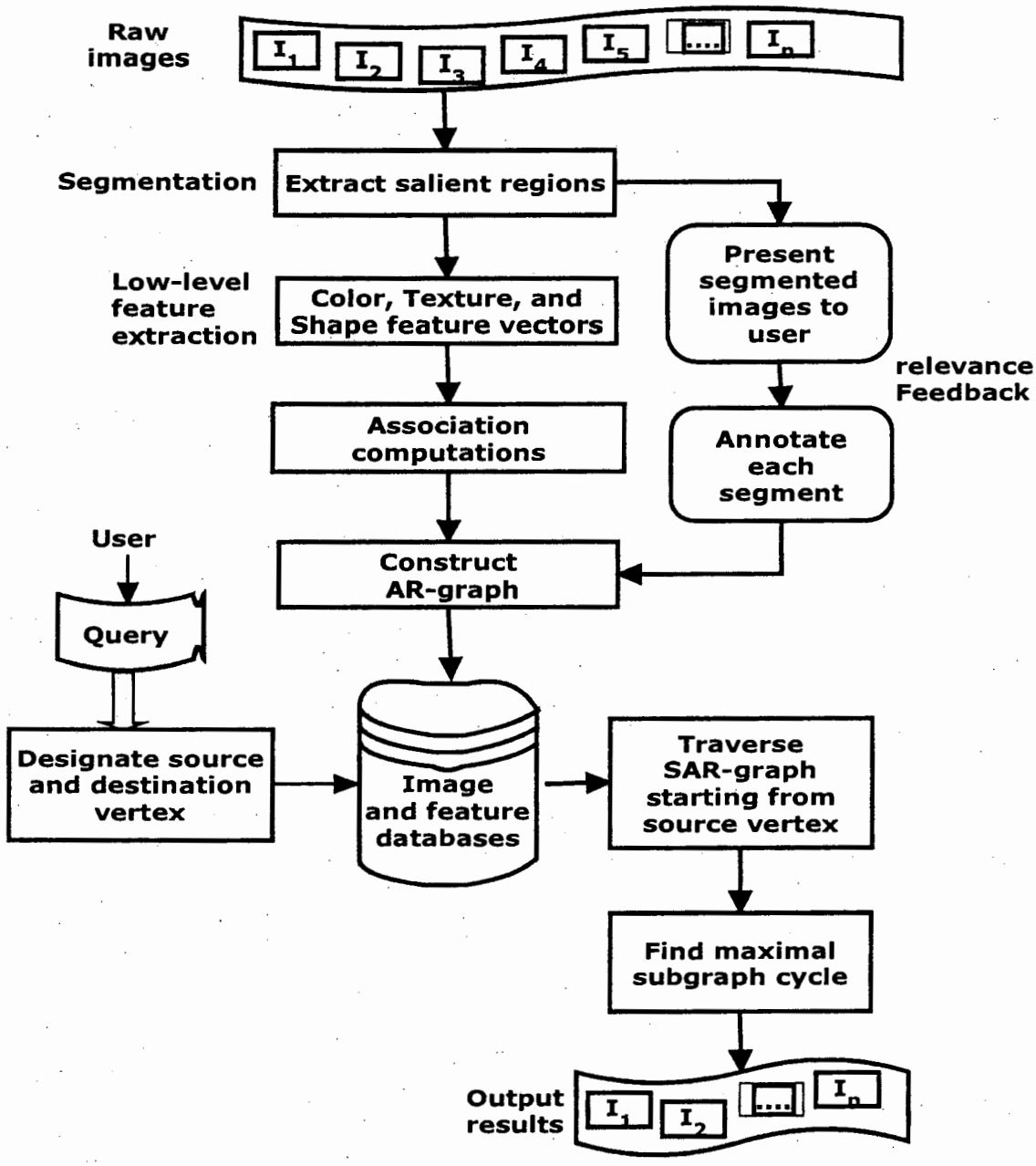


Figure 17: Proposed Semantic Image Retrieval system architecture

of images and annotations. This online annotation system allows researchers to segment and assign label to each image segment (object) and can share annotations with the rest of research community. The visual contents of the segmented regions can then be extracted

and stored in multi-dimensional feature vectors. These feature vectors of various segments form visual feature database.

To retrieve images, users provide the retrieval system with query image or keywords. The system then converts this query image into its internal representation of feature vectors. The similarities are then computed between the feature vectors of the query image and those of the images in the visual feature database. An AR-graph is then constructed using the *Association Ratio* as a proximity measure. The AR-graph provides an efficient way to retrieve similar images in the database. The proposed architecture also incorporates users' relevance early in the process that strengthens the retrieval process and to generate perceptually and semantically more meaningful retrieval results. The salient components of the system are described below.

Segmentation and feature extraction: Basically, raw images are presented in an iconic form as three-dimensional arrays of pixel values. Segmentation is a critical step of image analysis influenced not only by data but also by human factors. The segmentation component partitions images into local contents via region based method. Then, the feature extraction component extracts low-level features from each segmented regions. That is, each segmented region is represented by some visual feature vectors. We prefer manual tagging and segmentation as compared to automatic annotation and segmentation.

Manual annotation: Manual labeling is an important mechanism of including users' relevance early into image retrieval system. This module incorporates the users' perception of an image by assigning captions to each image segment before it is stored in the database. We have employed user centered segment based correspondence loop parallel to feature extraction module to capture human perception early in the system.

Image and visual features databases: This component contains all images and their corresponding visual features data. It may include the following:

– *Image database*—contains raw images

– *Visual Features database*—holds visual features extracted from raw images

– *Links database*—holds the connections between images and features

– *AR-graph*—contains the information as to how images are associated

Semantic image retrieval: measures the similarity between the query image and images in the database and constructs an ASD-tree considering one of the keywords as source vertex and another as destination vertex to find cycles in the AR-graph. All images corresponding to the tree nodes are retrieved as similar images.

User interface: this module presents a graphical user interface to let users interact with the system by submitting the queries and viewing the results.

5.3 Semantic Image Retrieval with Frequent Itemsets

Frequent itemsets (FI) have traditionally been used in business applications such as market basket analysis as introduced in Section 2.1. In this paradigm, items are the products in the market and transactions are any set of items purchased together. An FI is a set of items contained in the transactions with frequency greater than the user specified *min-sup* value. Since FI captures the association of similar transactions, the image retrieval based on FI is expected to perform well. The images grouped together to form an FI are semantically more similar to each other. This paradigm can discover similar images by capturing the semantic association among the semantically related images.

5.3.1 Test Image Collection

LabelMe (Russell et al., 2008) is an online image repository combined with a Web-based image segmentation and annotation tool. This tool provides a platform independent drawing interface and direct sharing of the datasets. LabelMe provides functionalities like i) segmentation of image objects by user drawn polygons; ii) assign labels to each segmented object in an image; iii) share/download whole or a portion of the database via the LabelMe Matlab toolbox. One can incorporate his/her own segmentation and annotation information in the database by just connecting control points along the edge of the object. The segmentation procedure is completed by clicking on the starting control point hence completing a polygon. Upon completion, a popup dialog bubble appears where a user can type in the label for newly created segment. Then, the feature extraction process can take image pixel values as input and produces the segment's visual features automatically or semi automatically. The extracted low-level features can be organized in the form of feature vectors.

A subset of images selected from the LabelMe database to be used for illustration purpose is shown in Figure 18. This sample dataset consists of 24 pairs of images, showing both original as well as segmented images. Each segment (object) is assigned a semantic label to describe image contents. These annotation labels are also shown under each image. These sample images cover a wide range of themes.

5.3.2 Frequent Itemset Framework for Image Retrieval

It is essential to model the image database in terms of items and transactions to employ FI framework for semantic image retrieval system. This model, called FI-based semantic image retrieval framework, is defined as follows:

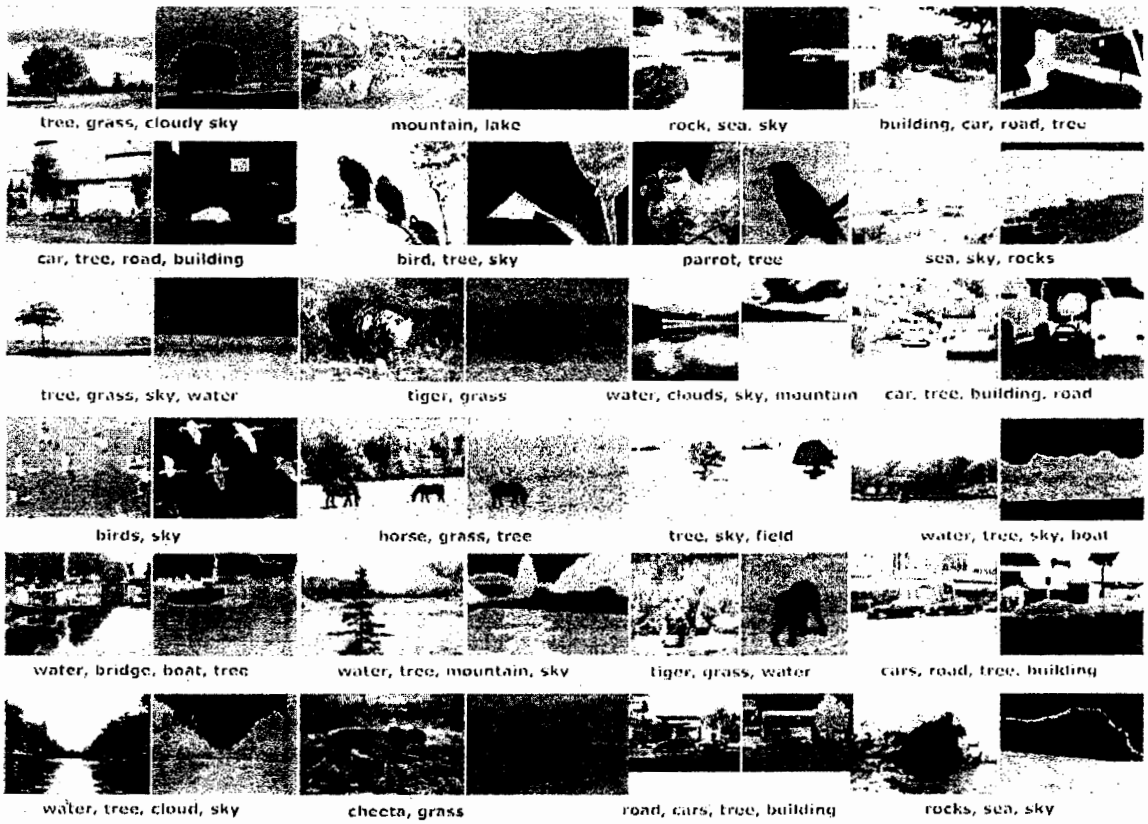


Figure 18: A set of example images along with segmentation and annotation selected from LabelMe database

- **Items**: The objects, segmented regions in an image, with correspondence
- **Transactions**: Set of images
- **Frequent itemsets**: Semantically related images

An image in the database corresponds to a market basket transaction and annotated objects in an image correspond to market basket items. Since an item can be a part of any number of transactions therefore, an object can arise in any number of images. A set of images are semantically similar if they contain objects of the same type. The objects in images are identified as homogeneous segmented regions with a meaningful label.

The task of semantic image retrieval in terms of frequent itemsets' framework can now be defined as;

- given at least a pair of keywords (items) the task is to find semantically similar images (transactions) containing same types of keywords (items) similar to the given keywords.

The output can then be ranked according to the association ratio weight i.e., an image containing more frequent regions get higher priority in the output list.

Table 13: Frequent itemset framework for the sample image dataset of

Image #	Objects
1	tree, grass, clouds, sky
2	mountain, water
3	rock, see, sky
4	building, car, road, tree
5	building, car, road, tree
6	bird, tree, sky
7	parrot, tree
8	sea, sky, rock
9	tree, grass, sky, water
10	tiger, grass
11	mountain, sky, clouds, water
12	building, car, road, tree
13	bird, sky
14	horse, grass, tree
15	tree, sky, field
16	water, sky, boat, tree
17	water, boat, tree, bridge
18	water, tree, sky, mountain
19	tiger, grass, water
20	building, car, road, tree
21	water, tree, clouds, sky
22	cheeta, grass
23	building, car, road, tree
24	rocks, sea, sky

For the images $\{I_1, I_2, I_3, ..., I_{24}\}$ in Figure 18, the frequent itemset framework corresponding to these images is shown in Table 13. The table has two columns: column one corresponds to image identities I_i's, and second column represents the segment labels i.e., $\{w_1, w_2, ..., w_{19}\}$ = {*bird, boat, bridge, building, car, cheetah, clouds, field, grass, horse, mountain, parrot, road, rock, sea, sky, tiger, tree, water*}.

5.3.3 Semantic Association Measurement

According to the FI framework, each image is regarded as a transaction containing a set of items (labels). Usually, an image is represented as vectors of m features and attributes. Each image region is represented by m-dimensional feature vector of n features. The attributes are set-valued, such as the captions of the image regions. Set-valued attributes have different number of elements, and there is no given alignment between set elements. However, where the set representation and vector representation isolates each feature, the graph allows us to combine valuable information *between* individual features and attributes. Therefore, graphs are used as a straightforward representation in this framework. Hence, the retrieval performance could be improved if the visual feature information is incorporated in the form of graph vertices and edges. Using the freedom in defining the graph, varied information can be added as attributes to graph vertices and edges.

A similar approach to FI can be applied to measure the semantic similarities between image labels by treating each image as market basket transaction containing the labeling keywords as items. The proposed approach is described as follows. Each image is segmented to identify salient objects and annotated with keywords. The *association ratio* metric, introduced in Section 4.1.2, is an effective measure to find the association among

the keywords using annotation co-occurrence matrix (Table 14). The annotation co-occurrence matrix $M_{|D|\times j}$, describes the relationship between the labels and the images, where the i^{th} row denotes the annotation of image I_i, the j^{th} column denotes the pattern of labels w_i occurring in the dataset, and M_{ij} denotes whether the i^{th} image contains the annotation of the j^{th} label as defined in the equation 5.1.

$$M_{ij} = \begin{cases} 1, & \textit{if } j^{th} \textit{ label exists in } i^{th} \textit{ image} \\ 0, & \textit{otherwise} \end{cases} \qquad (5.1)$$

Now these co-occurrence patterns can be used to compute the probability of co-occurrence between any pair of keywords. The co-occurrence probability between w_a and w_b is defined as follows:

$$P(w_a, w_b) = \frac{\sum_{j \in D} M_{ja} \times M_{jb}}{|D|} \qquad (5.2)$$

For measuring semantic association we have *association ratio* (AR) defined as follows:

$$R_{(w_a, w_b)} = \frac{P(w_a, w_b)}{(1 - P(w_a, w_b))} \qquad (5.3)$$

Eqn. 5.3 measures the association ratio between the two keywords w_a and w_b. The larger association ratio value indicates more semantic correlation and smaller values represent less correlation. This is due to the frequent co-presence of objects in a similar context. A value of zero i.e., absence of an edge means no correlation at all.

For illustration, there are 19 keywords in the annotation vocabulary V = {bird, boat, bridge, building, car, cheetah, clouds, field, grass, horse, mountain, parrot, road, rock, sea, sky, tiger, tree, water}, and 24 images in the example dataset. The annotation co-occurrence matrix is defined in Table 14. The semantic association between the keywords "sky" and "tree" measured by Equation 5.3 is 0.40. Similarly, the association

ratio between "boat" and "water" is calculated as 0.09 which is low due to their low co-occurrence frequency in the dataset.

Table 14: Annotation co-occurrence matrix

Image #	bird	boat	bridge	building	car	cheetah	clouds	field	grass	horse	mountain	parrot	road	rock	sea	sky	tiger	tree	water
1							1		1							1		1	
2											1								1
3														1	1	1			
4				1	1								1					1	
5				1	1								1					1	
6	1															1		1	
7												1						1	
8														1	1	1			
9									1							1		1	1
10									1								1		
11							1				1					1			1
12				1	1								1					1	
13	1															1			
14									1	1								1	
15								1								1		1	
16		1														1		1	1
17		1	1															1	1
18											1					1		1	1
19									1								1		1
20				1	1								1					1	
21							1									1		1	1
22						1			1				1					1	
23				1	1														
24														1	1	1			

5.3.4 Image Features

The most common image feature extraction mechanism is to use the pixel values as features directly. This quantitative information is the basis for differentiating one class of images from another. The multiple semantics related features called visual signatures are extracted to obtain as much semantic contents as possible. In the feature based approach the input image I is described by features that are arranged in feature vectors as

$$f(I)=(f_1, f_2, ..., f_n) \qquad (5.4)$$

For a given dataset of annotated images D, let $|D|$ denote the size of D. Each annotated image I_i in the dataset can be expressed using a set of image regions (objects) and annotation words as

$$I_i = \{f_{1,1}^{(i)}, f_{1,2}^{(i)}, \ldots, f_{m,n}^{(i)}, w_1^{(i)}, w_2^{(i)}, \ldots, w_p^{(i)}\} \qquad (5.5)$$

where m is the number of objects contained in an image, n represents the number of features (color, texture, and shape) of each object and p is number of annotation words. The color feature is represented as a color histogram based on the L*a*b* color space (Russ, 1998), where L represents the luminance axis, which is equivalent to the grey axis. The coordinates (a, b) represent the chrominance, considering a constant luminance. The axis a corresponds to the antagonist pair green-red and the axis b corresponds to the pair yellow-blue. This color space works well in perpetual uniformity and provides very good estimates of color difference between two color vectors, because this color difference is calculated with the Euclidean distance.

The L*a*b* *color* space is quantized into 64 bins (4 each for L, a, and b) to reduce processing overhead. Thus a 64 dimensional feature vector C_i is obtained for each image as a color feature representation. The computational complexity of histogram is linear with respect to image size.

Texture features represents the characteristics such as the smoothness, regularity, and coarseness of an image. We use Gabor (Manjunath and Ma, 1996) filter to characterize the underlying texture information of an image. This filter is designed in such a way that Fourier transformations of filters cover most of the image spectrum. Given an image $I(x, y)$, its Gabor wavelet transform is defined as

$$I_{mn(x,y)} = \int I(x,y)\, g_{mn}^{*}(x-s, y-t)\, dx\, dy \qquad (5.6)$$

where s, t are dimensions of filter and g^{*}_{mn} is the complex conjugate of the Gabor wavelet. Texture feature vector is constructed as the mean μ_{mn} and the standard deviation σ_{mn} of the magnitude of transform coefficients. The feature vector of image I is defined as $T_{gb}(I) = [\mu_{00}, \sigma_{00}, \mu_{01}, \sigma_{01}, \ldots, \mu_{mn}, \sigma_{mn}]$. Applying the Gabor filter to every image pixel (x, y) results texture features represented as 35-D vectors.

Shape is another binary image feature consisting of contour or outline of objects, obtained after segmentation. Shape feature representations are either boundary-based or region-based. Boundary-based technique describes the shape region by using its external characteristics, for example pixel along the object boundary, while the region-based technique describes the shape region by using its internal characteristics, for example the pixel contained in the region.

Subsequently the similarity of image is measured as distance of feature vectors using L(2) norm

$$D(I,Q) = \left| \sqrt{\sum_{i=1}^{n} \left((I_{ik})^2 - (Q_k)^2 \right)} \right| \qquad (5.7)$$

where

D is the Euclidean distance

I is the database image feature vectors

Q is the query image feature vector

$i = 1, 2, .., n,$ and n is the total number of images in the database

$k = 1, 2, \& 3$ are the indices for color, texture, and shape features

The resultant distance computed between the database images (I) and query image (Q), if k numbers of feature vectors are considered for retrieval, k different distances

$d_{ki}(I_i(x,y),Q(x,y))$ will be obtained. The effective object feature distance obtained from the weighted sum of each feature distance is given by

$$D_{mn} = \sum_i R_i \, d_{ki}(I_i(x,y),Q(x,y)) \qquad (5.8)$$

where R_i is association ratio as computed in Equation 5.3.

The exact feature extraction details are orthogonal to our approach. Any method that incorporates primitive features to form a feature vector is supposed to meet our system's requirements. Since image features $f \in \Re^n$, it is required to quantize the feature vectors so that the visual signatures are indexed efficiently. In this work, a visual feature database is created containing each feature attribute to accomplish this objective.

5.3.5 Visual Features Database

The creation of visual feature database (VFD) is a fundamental pre-processing step to organize visual features. A valid retrieval or annotation framework is difficult to build without organizing similar objects. Each element of VFD represents the mean of object feature groups. Without VFD we would have to consider all feature values resulting in a big performance hindrance due to huge feature values.

Our object, in this work, is not to design a new indexing method for high-dimensional image features but to use an existing one efficiently and effectively. Each normalized segmented region is described with a SIFT-descriptor (Lowe, 2004). Then VFD is built by clustering the SIFT-descriptors with the hierarchical-agglomerative method explained in (Leibe and Schiele, 2003). The similarity based feature grouping is achieved by applying this procedure to each feature attribute and hence creating a VFD for all objects.

5.3.6 Semantic AR-graph

Although one can take the advantage of WordNet semantic sub-graph as external background knowledge, it is difficult to handle misinterpretation problem coming from the different senses of a term. To address the issue, it is proposed to build a semantic graph to clearly illustrate the semantic association among the semantically related set of images. The local semantic relationship among the labeled image objects can be extracted using frequent itemset (FI) framework. Given that the images and annotated objects (segmented labeled homogeneous regions), constructing an undirected weighted AR-graph involves making each object as a graph vertex and co-occurrence of two objects in an image corresponds to creation of an edge between the two corresponding vertices. A pair of image objects is connected by an edge to incorporate semantic association information in the graph. The association is represented as an edge weight and corresponding image information is stored as pointers to images containing the segmented pair. The model feature structure of semantic AR-graph is shown schematically in Figure 19.

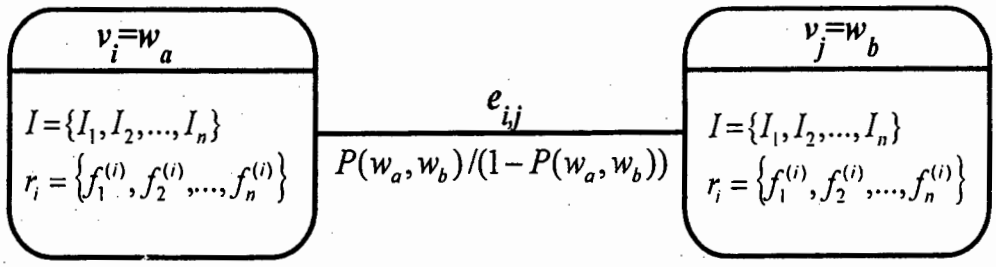


Figure 19: A model of semantic AR-graph

Each segmented region of an image is represented by an AR-graph vertex. This reduces the number of vertices in the graph, which results in performance gain. The caption

assigned to the segment is used as label for the vertex. Segmentation produces many independent regions, $r_1, r_2, \ldots, r_k$, from an image. Two segmented regions r_i and r_j occurring simultaneously in an image result in the creation of an edge $e_{i,j}$ between the two vertices. The edge attributes describe the relationship between the two vertices (segments). The edge weight represents the semantic association of segments in the image database. We now have Algorithm 4 for the construction of semantic AR-graph below.

/* **Algorithm 4**: Construction of semantic AR-graph */

Input: *Image dataset* ***D*** *containing* ***n*** *images*
Output: *Semantic AR-Graph* $G = (V, E)$

1. Set $V(G) = \Phi$ and $E(G) = \Phi$
2. **for each** image $I_1, I_2, \ldots, I_n \in D$ **do**
3. Segment the image into **k** homogeneous regions
4. **for each** Segment(i) = 1 to k /* where k = |segments| */
5. v_i = SegmentLabel(i)
6. **if** $v_i \notin V(G)$
7. $V(G) = V(G) \cup v_i$
8. $V(G) = V(G) \cup I_i$
9. **end**
10. **for** $j = i+1$ to k
11. **if** $v_j \notin V(G)$
12. $V(G) = V(G) \cup v_j$
13. $V(G) = V(G) \cup I_j$
14. **end**
15. **if** $(e_{i,j}) \notin E(G)$ /* edge between vertex v_i and v_j */
16. $E(G) = E(G) \cup e_{i,j}$
17. **end**
18. $Weight(e_{i,j})$++
19. $$Weight(e_{i,j}) = Weight(e_{i,j}) \frac{\sum_{j_i \in S} M_{ja} \times M_{jh}}{|S|}$$
20. **end**
21. **end**
22. **for each** edge $E_{i,j} \in G$
23. $Weight(e_{i,j}) = Weight(e_{i,j}) + P(e_{i,j})/(1 - P(e_{i,j}))$ /* where $P(e_{i,j}) = f(x_i, x_j)/|D|$ */
24. **end**
25. **end**

Analysis: Supervised learning models usually require a training step on an annotated database called training dataset. Constructing such types of datasets is not a simple process. Moreover, such models are complex and computationally very costly to train. Our proposed framework does not require any training phase and also no parameters to be tuned. By relating image segments via correspondence, semantic AR-graph avoids detailed specifications of concepts or complicated similarity measures.

Figure 16 shows a semantic AR-graph drawn for the example dataset given in Figure 14. The image captions are used as the graph vertices and the edge weight between the two vertices is computed using association ratio metric. This is evident that the semantic closeness among the labels is measured explicitly and absence of correlation is shown as absence of an edge. For example, the labels "sky" and 'tree" are more semantically correlated than "sky" and "mountain". These semantic associations measured among the object captions agree well to the subjective perceptions of the image contents.

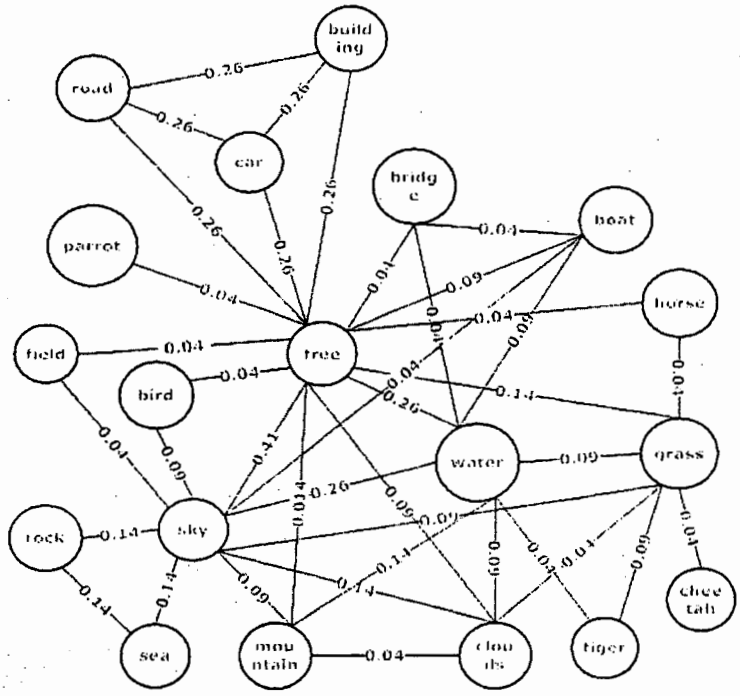


Figure 20: Semantic AR-graph for the sample dataset

5.3.7 Semantic Image Retrieval System

Semantic image retrieval combines many different fields of research. Obviously this work cannot cover all of them in detail. The emphasis is laid on the efficient extraction and matching of visual image features incorporated in an AR-graph. Implementation of the semantic image retrieval system (SIRS) is a complicated process. In the first step this was observed that images in the image dataset were not of the same size. All images were resized to 256 x 256 dimensions because LabelMe contained image dimensions of as much as 1200 x 1600. Reducing the image size below 256 x 256 may lead to distortion (blurring) of the images and may give poor retrieval results. On the other hand the images need not be resized beyond 256 x 256 because the system is expected to be computationally efficient.

Initially, annotated and segmented images are loaded into the system. With the visual feature database ready, an order for the object labels is calculated and an index to each label is allocated. Given an image, for each feature attribute, it is replaced with the index of the label to which it is assigned in the corresponding VFD. Then low-level features of segmented objects are extracted, while keeping discrete value type for each attribute in a limited domain. To build semantic AR-graph each object label corresponds to graph vertex and co-occurrence of any two objects in an image corresponds to creation of an edge. Each edge weight is computed using frequent objects association ratio and visual features attribute values. Associated with each node (vertex) of the semantic AR-graph are pointers to the images containing the object correspond to the node.

An important functionality offered by the system is integration of semantic AR-graph with the visual feature database as a background knowledge base. An image retrieval

algorithm is designed around the semantic AR-graph using the visual feature database for the images. In this algorithm, similar images are retrieved for the query image by traversing the semantic AR-graph to construct ASD-tree. When query keywords are received by the system one of the keyword is designated as source w_s and another as destination w_t. Similar images are retrieved using the semantic AR-graph traversal for the query keywords by constructing as ASD-tree making w_s as the root node and w_t as destination node. All the nodes in the source-to-destination paths are selected. The images corresponding to the path nodes are retrieved and produced as similar images. The images retrieved in this way can be ranked by computing the sum of all edge weights along the path in ASD-tree. Since this method is independent of similarity measures thus any similarity metric can be used for various application domains. In this work, Euclidian similarity measure is used for measuring query object and VFD similarity due to its simplicity and effectiveness.

Since the algorithm constructs the ASD-tree by visiting only the vertices in the source to destination vertices, the image retrieval is efficient and scalable to the database growth. With this approach, images can be accessed not only based on the low-level feature similarity but also using semantic association among the keywords in the form of semantic AR-graph. Therefore, this image organization and retrieval mechanism fulfills the objectives listed earlier. The search algorithm is given as follows:

Algorithm 5 Finding similar images using semantic AR-graph.

Input: *A semantic AR-graph; Keywords* $w_1, w_2, \ldots, w_n$
Output: *Similar images* $I_1, I_2, \ldots, I_k$

1. *Initialize ASD-tree* $T = (V, E)$ *by setting* $V(T) = \Phi$ *and* $E(T) = \Phi$
2. *Take two keywords designating one as source* (w_s) *and another as destination* (w_t)
3. *Root node of* $T = w_s$
4. $i = 1$
5. ***do while*** (*all tree nodes are not traversed*)
6. $t_{[1, 2, \ldots, n]}$ = *get all vertices reachable from* t_i *in AR-graph*
7. ***for each*** $t_j \in t_{[1, 2, \ldots, n]}$
8. ***if*** (t_j != w_t && t_j *does not exist in the path from* t_i *to root*)
9. *Add* t_j *as* t_i*'s successor*
10. ***end if***
11. ***end for***
12. i++
13. ***end do***
14. *Retrieve all images corresponding to each node of ASD-tree paths*
15. *Rank the result in order of sum of edge weights along the corresponding paths*

5.4 Experiments and Results

The evaluation of an image retrieval system is a complex task because there are variety of measures for quantitative performance analysis. Moreover, collection of really large image databases with predefined ground truth does not exist yet. It is also difficult to define a ground truth as different users may consider different kinds of similarities such as semantic similarity, visual similarity etc. Further, it may also be difficult to have a precise definition of similarity. There may exist images, that may appear somewhat similar according to the low-level visual features, but their degree of semantic similarity is very low. How should a measure respond to such images if they are found and/or not found? As a result, different image retrieval systems can hardly be compared. Nevertheless a quantitative analysis is helpful to evaluate improvements within one system. We therefore compare evaluation measures for queries with different parameter sets. The effectiveness and efficiency of the proposed framework can be examined by performing an extensive experimental evaluation. Here, we evaluate the proposed

approach on a publically available research database called LableMe. The algorithms are implemented in Matlab. All the experiments are performed on Windows XP in a 3.2 GHz IBM compatible PC with 1 GB of memory.

5.4.1 Dataset

When evaluating the performance of different algorithms for semantic image retrieval, segmented annotated database is essential to quantitatively compare their performance. Large datasets of segmented annotated images with many objects present in an image are very rare. Usually, datasets are constructed and tailored to retrieve, classify, or recognize a specific object by a particular research group. Majority of datasets are built to solve a specific problem such as object classification. Therefore, many present day datasets only consist of a small number of object classes, such as cars, buildings, or people etc. However, few datasets contain segment annotation information, because it is difficult to create such datasets. The Caltech-256 dataset (Griffin, Holub and Perona, 2007) with 256 object classes, the PASCAL (Everingham et al., 2005) object recognition database collection contain five fully annotated, three partially annotated and one unannotated database, and the CBCL-streetscenes database (Bileschi, 2006) are few such efforts by the research groups.

LabelMe (Russell et al., 2008) is a latest addition in the list of datasets of segmented annotated images with many diverse objects in the same images. The dataset covers a wide range of locations and a number of object categories. A user can create his/her user name and can logon to the system by the user name. This helps LabelMe to recognize users for the objects he/she has assigned labels during the labeling session. The labels are then readily available for use and visible during subsequent visits of the same image.

5.4.2 Performance Evaluation

The content of the LabelMe (Russell et al., 2008) dataset are summarized as of March, 2009, which consists of 159496 images out of which 44807 images are annotated. The data are organized into folders and the folder names providing information about the image contents and location of the depicted scenes. There are two types of folders: i) static pictures and, ii) video sequences. Each video frame can be treated as independent still image. The source of annotation is either LabelMe online annotation tool or annotation tools developed by other research groups.

The image retrieval evaluations were performed on a general purpose color image collection containing 1200 fully annotated images selected from LabelMe database. The selected images are not all visually alike and contain at least five objects and each object having assigned a suitable caption. The relevancy of the results was evaluated by the users and the retrieval accuracy was the average values across all query sessions. The merit of this method is that the association between captions and the segment based image feature generation estimation can both be readily combined in a retrieval algorithm.

In order to show the validity of the method, the experiments were designed to demonstrate the overall effectiveness of the proposed algorithm in improving the image retrieval. The *precision* and *recall* (Smith and Chang, 1996) are the common evaluation measures used in SIRS are defined as:

$$\text{Precision} = \frac{\text{Number of relevant images retrieved}}{\text{Total number of images retrieved}}, \qquad 5.9$$

and

$$\text{Recall} = \frac{\text{No of relevant images retrieved}}{\text{Total number of relevant images in the database}}. \qquad 5.10$$

The precision measures the accuracy of the retrieval system and recall measures the completeness of the retrieval system. The average precision and recall evaluates the overall system performance.

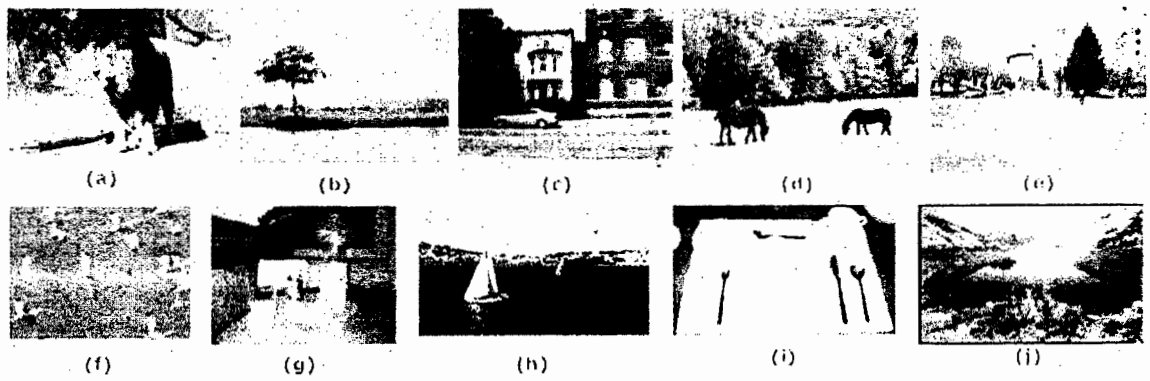


Figure 21: Query images selected from the database

Figure 21 displays query images (keywords) used for the experiment. For each image all similar images within the database were selected manually thus defining a ground truth for each query image. The selection of similar images was not motivated from object features, instead the images were drawn based on similar visual appearance.

During precision-recall tests one hundred images (top-10 images for each query image) were retrieved. The average retrieval performance of this system was examined by computing precision and recall for each query operation. The retrieval results were evaluated with and without employing semantic AR-graph. Figure 22 shows the plot of precision and recall for the 10 query images (keywords). Figure 22 also shows the plot of average precision and recall after retrieval of number of relevant images. Results are shown for the top-10 images retrieved by the system.

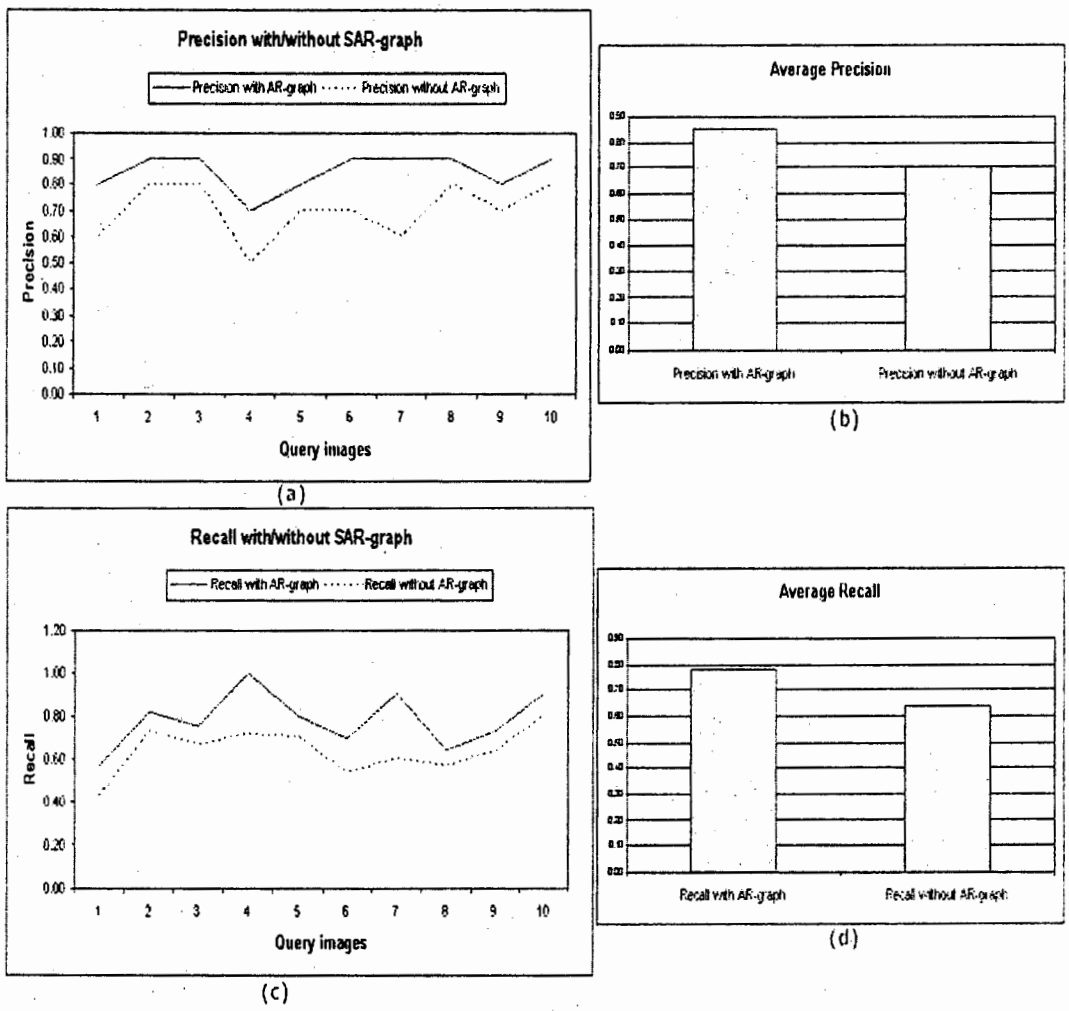
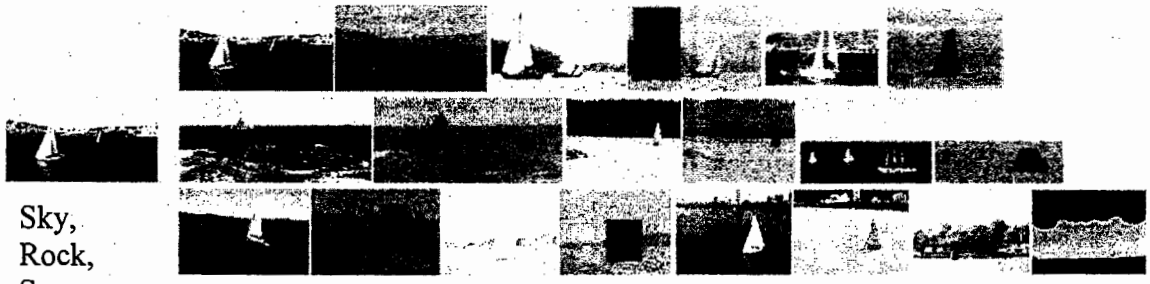


Figure 22: Precision and Recall comparison with / without Semantic AR-graph

In this approach, multiple operations, such as segmentation, feature extraction, annotation, and feature indexing have been integrated to organize the visual feature components, and constructing more flexible visual features for the efficient semantic retrieval of the images. This improves the performance of retrieval system significantly when large number of labeled images is available in the database. Thus, our approach shows predictably better performance as compared to traditional methods and their recent variants. It is observed that incorporating semantic AR-graph improved the retrieval process significantly.

5.4.3 Retrieval Results

This approach not only retrieves semantically related images based on the low-level visual feature similarity but also using the semantic association between frequent image objects by means of the semantic AR-graph. The top-10 relevant images retrieved in each query session are determined by the corresponding visual feature processing accuracy and its semantic graph model. In addition to the quantitative plots given above we present some retrieval results visually. For the presented query images of Figure 21 which had shown the overall best performance in the quantitative analysis.



Sky, Rock, Sea, Yacht

Figure 23: Query results for the image of Figure 21-h

Figure 23 presents an example query for a "sky", "rock", "sea", and "yacht". The query images were selected from the test image dataset. The system correctly returns the original image on the first rank and several other semantically related images ranked next from left to right and top to bottom. However, there exist a case (last image) for which the color and texture has more effect on the retrieval process.

5.5 Summary

In this chapter, we have proposed a data mining approach to model the associations among the semantically related images. A semantic association ratio based structure,

called semantic AR-graph, is presented to capture the semantic relationship. Based on this approach object captions are linked with each other showing the corresponding semantic association as edge weights. After generating multiple low-level image features and stored in visual feature database (VFD), the proposed architecture utilizes this for the semantic image retrieval problem. A unique semantic image retrieval algorithm is designed by integrating the semantic AR-graph and low-level image features. The experimental evaluations have demonstrated that the proposed method models the semantic association effectively and the image retrieval prototype system utilizing the derived model is promising both in effectiveness and efficiency.

CHAPTER 6

DISCUSSION

We have developed minimum support free algorithms for effective and efficient mining of frequent patterns and semantic image retrieval tasks. Now we first summarize the most important characteristics of our approaches and then discuss their extensions and applications.

6.1 Characteristics

We have developed a novel, effective and efficient frequent pattern mining method that does not require setting of min-sup threshold by the user. Major characteristics of the method are summarized as follows;

- This method for mining frequent patterns does not require *min-sup* parameter to be provided by the user. This allows true exploratory data mining by relieving users of ascertaining basic characteristics of underlying data.
- The cornerstone of this approach is an AR-graph, a compact data structure, reduces number of database scans to one and provides time and space efficiency.
- An all path source to destination tree (ASD-tree) is constructed on top of AR-graph to retrieve MFIs corresponding to the graph cycles. The weight computed as the sum of all edge weights along the source to destination path is used to rank the results.

- Generally data mining algorithms have to search very large databases. Our method first converts horizontal or vertical input data into a compact graph-based representation. This enables the search algorithm to focus on reduced and compact representation hence reducing the search space.
- This approach avoids candidate generation and testing altogether. Instead of bottom-up approach, as the majority of candidate generation methods adopt, this method uses top-down approach. This makes our method very efficient since the number of candidate sets generation in bottom-up approach could be huge.
- Graph-based method avoids most of expensive pattern matching operations. This produces maximal frequent itemset and *top-k* maximal frequent itemsets directly, which is very much cheaper.
- The parameter-free methods can compete with and outperform parameter-laden methods on a vide variety of application domains.

6.2 Application to other Data Mining Tasks

Since the first introduction of this technology, i.e., frequent pattern mining, it has become a primitive mechanism in many data mining applications. With over a decade of extensive research work frequent pattern mining has been applied to various kinds of extensions and applications, ranging from scalable data mining methodologies to handling a wide diversity of data types, various extended mining tasks and a variety of new applications. It is understood that this framework can also be extended to other data mining tasks as well.

6.2.1 Mining Multilevel Association Rules

This work can naturally be extended to mining multilevel association rules. A multilevel association rule involves concepts at different levels of abstractions. In many real world domains, it is hard to discover strong association among data items at low level of concepts due to the sparsity of data. Alternatively, interesting associations representing commonsense knowledge can be derived at higher levels of abstraction. Therefore, multilevel association rules can be mined efficiently by combining AR-graph with concept hierarchies. Mining frequent itemsets without *min-sup* threshold provides an efficient framework because *min-sup* changes at different concept levels such as data cubes in data warehouse.

6.2.2 Mining Complex Patterns

The structures like graphs, trees, and lattices contain more complicated patterns that can be found in many scientific and commercial applications such high-throughput screening, accelerated drug development, etc. Graph data structures have generally become significant in modeling complex structures such as bioinformatics, image retrieval and indexing, computer vision, Web analysis and text processing. Mining frequent substructures in a collection of graphs is another important application. Finding frequent substructures means to discover sub-graphs contained in many attributed graphs. Frequent sub-graphs can be mined by finding maximal frequent item tours in an AR-graph.

6.2.3 Frequent Pattern-based Clustering

Clustering is another important fundamental concept of data mining which can be defined as the task of grouping together of similar data items called clusters. Clustering has been

applied to a wide range of application areas. Distance functions are hard to define properly for multimedia databases that often have very high dimensionality. Frequent pattern mining is a promising candidate technique for such problems that contain very high dimension data. Objects can be grouped into some clusters according to the patterns they share after finding a set of frequent patterns. This can help us in avoiding the problem of defining distance functions and dealing with high dimensionality explicitly.

6.3 Summary

The methods presented in this work find frequent patterns using an association ratio graph. They use effective and simple algorithms to extract *top-most* and *top-k* maximal frequent itemsets. Our study shows that mining without *min-sup* threshold parameter is not only effective but also efficient. This has strong implication to mining other kinds of knowledge and broad applications such as clustering and classification.

CHAPTER 7

CONCLUSION

Most of the current frequent pattern mining techniques have taken for granted that users can specify the *min-sup* value easily. But this is not the case. The performance of support-based algorithms is heavily dependent on the user specified *min-sup* threshold even though it is fine-tuned under the supervision of an experienced data analyst. Therefore, users are unnecessarily burdened to know the characteristics of the underlying database in order to specify appropriate *min-sup* threshold. It is understood that specifying the suitable threshold is quite subtle, which can hamper the wide-ranging applications of frequent pattern mining methods. Existing methods for removing or resolving the *min-sup* are not fully developed. Some approaches try to eliminate *min-sup* by mining only the top 10 or 20 percent of the prospects with the highest score. Others call for mining top-*k* frequent itemsets for effectiveness and efficiency. However, majority of the existing approaches are still dependent on computation of *min-sup* threshold directly or indirectly.

In this work, we propose a novel parameter-free frequent pattern mining framework based on AR-graph structure that allows true exploratory data mining. We have made a good attempt towards removing *min-sup* parameter from the mining process. It is shown that the frequent pattern mining algorithms based on *min-sup* are burdensome to use and

parameter-free algorithms are realizable for this particular class of problem. This approach can limit users' capacity to force his/her prejudices on the problem at hand and allows true exploration of the data. It is also demonstrated that this technique is competitive with the state-of-the-art methods. Apart from market basket problem it can also be applied to other challenging data mining tasks such as semantic image retrieval. In this chapter, we first summarize our work and then discuss some interesting future directions.

7.1 Summary of the work

The task of discovering maximal frequent itemsets without *min-sup* threshold is quite challenging. Finding frequent itemsets is vital in mining correlations, associations, classification, clustering, and many other data mining tasks as well. In this work, we propose parameter-free graph-based algorithms for mining *top-most* and *top-k* maximal frequent itemsets making following contributions.

- A novel AR-graph based structure for compact representation of transaction the database is proposed. The presence of an edge between a pair of vertices indicates an association and absence of an edge represents no association. Further, all edges are assigned weights computed using a novel *association ratio* metric showing the level of association between vertices. Larger edge weights show greater association among the vertices (items) and vice versa. An AR-graph can easily be represented in computer memory using a sparse symmetric matrix.
- This compact AR-graph representation is helpful in the development of an efficient mining method for finding maximal frequent patterns. Efficiency is

achieved by: i) converting a large database is into a highly compact format that avoids costly repeated database scans, ii) adopting a top-down approach thus avoiding candidate generation and testing, and iii) applying breadth-first search approach to discover maximal frequent item tour (cycle) for mining maximal frequent itemsets in large databases, which reduces the search space considerably.

- The major cost of the method is that it has to build an undirected weighted graph and compute association ratio for the graph edges. This method has shown high performance especially for sparse datasets.
- The study shows that our approach is effective and scalable for mining both long and short frequent patterns, and is also efficient as compared with some of the existing mining methods.
- Majority of existing methods do not provide any control to users to derive compact and high quality itemsets, which are useful in many applications. The proposed method reduces such a huge set by finding only *top-most* and *top-k* maximal frequent itemsets without user specified *min-sup* threshold.
- Bridging the semantic and sensory gaps for semantic image retrieval are very hard problems to tackle. Major difficulty in this context is to make computers understand image contents in terms of high-level concepts, which is closely related to the problem of computer vision and object recognition. A systematic framework is proposed to map the frequent pattern approach for efficient retrieval of semantically similar images in large multimedia databases. This new

approach adopts a maximal sub-graph cycle principle, by extending the AR-graph based algorithm to retrieve semantically similar images.

7.2 Future Research Directions

Our agenda for future work consists of further work to develop pattern-based mining methods. For example, classification is an important data mining task. In classification each tuple belongs to a class among a pre-defined set of classes. A user-specified goal attribute decides the class of a tuple. Is it possible to build better classification models using frequent patterns than most other classification methods? Which type of frequent patterns is more useful than other frequent patterns? Can such patterns be extracted directly from data? These questions need to be answered before frequent patterns can play an essential role in several major data mining tasks, such as classification.

It is also required to analyze possible applications for the deep understanding and interpretation of patterns, e.g., semantic annotation for frequent patterns, and contextual analysis of frequent patterns. The main thrust of pattern analysis research work has been focused on pattern composition and frequency. The semantic of a frequent pattern includes further information i.e., what is the meaning of the pattern; what are the synonym patterns; and what are the typical transactions that this pattern contains?

7.3 Final Thoughts

"*Science is what you know; philosophy is what you don't know*[3]". Data mining provides effective and efficient tools that can turn *what you don't know* (philosophies) to *what you know* (science). Frequent pattern mining as a young research field has achieved

---

[3] *Encarta® Book of Quotations* © & (P) 1999 Microsoft Corporation. All rights reserved. Developed for Microsoft by Bloomsbury Publishing Plc.

tremendous progress and claimed a nice set of application domains. It is believed that the frequent pattern mining research has widened the extent of data analysis considerably. In the long run it will have deep impact on data mining tasks, methodologies and applications. However, it still remains the responsibility of the data miner to distinguish the gold from the dust.

Bibliography

1. Agarwal R., Aggarwal, C.C., & Prasad, V.V.V. (2000). Depth first generation of long patterns. In: *ACM Conf. on Knowledge Discovery and Data Mining (SIGKDD)*, pp 108-118

2. Agrawal, R., Imielinski, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proc. 1993 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'93)*, pages 207–216, Washington, DC

3. Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. In: *Proceedings of the 20th International Conference on Very Large Databases (VLDB'94)*, Santiago, Chile. Morgan Kaufmann, pp 487-499

4. Agrawal, R., & Srikant, R. (1995). Mining sequential patterns. In *Proc. 1995 Int. Conf. Data Engineering (ICDE'95)*, pages 3–14, Taipei, Taiwan

5. Baeza-Yates, R., & Ribeiro-Neto, B. (1999). Modern Information Retrieval. *ACM press*, 123-129

6. Bayardo, R.J. (1998). Efficiently mining long patterns from databases. In: *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*

7. Bileschi, S. (2006). CBCL streetscenes. Technical report, MIT CBCL, 2006. The CBCL-Streetscenes dataset can be downloaded at http : //cbcl.mit.edu/software-datasets.

8. Borgelt, C. (2009). Christian Borgelt's Webpages, accessed on March 3rd, 2009, http//www.borgelt.net

9. Brin, S., Motwani, R., Ullman, J., & Tsur, S. (1997). Dynamic itemset counting and implication rules for market basket data. In: *Proceedings of ACM SIGMOD international conference on management of data*, pp 255–264

10. Burdick, D., Calimlim, M., & Gehrke, J. (2001). Mafia: a maximal frequent itemset algorithm for transactional databases. In: *IEEE Intl. Conf. on Data Engineering (ICDE)*, Heidelberg, Germany

11. Carson, C., Belongie, S., Greenspan, H., & Malik, J. (2002). Blobworld: Image segmentation using expectation-maximization and its application to image querying. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 1026–1038

12. Cheung, Y.L., & Fu, A.W. (2004). Mining association rules without support threshold: with and without item constraints. In: *TKDE*

13. Chuang, K., Huang, J.L., & Chen, M.S. (2008). Mining top-k frequent patterns in the presence of the memory constraint. In: *The VLDB Journal* 17:1321-1344

14. Chung, S.M., & Luo, C. (2008). Efficient mining of maximal frequent itemsets from databases on a cluster of workstations, *Knowledge Information System* 16:359–391

15. Datta, R., Joshi, D., Li, J., & Wang, J.Z. (2008). Image Retrieval: Ideas, Influences, and Trends of the New Age. *ACM* 1529-3785/2008/0700-0001

16. Dong, G., & Li, J. (1999). Efficient mining of emerging patterns: Discovering trends and differences. In *Proc. 1999 Int. Conf. Knowledge Discovery and Data Mining (KDD'99)*, pages 43–52, San Diego, CA

17. Everingham, M., Zisserman, A., Williams, C., Van Gool, L., Allan, M., Bishop, C., Chapelle, O., Dalal, N., Deselaers, T., Dorko, G., Duffner, S., Eichhorn, J., Farquhar, J., Fritz, M., Garcia, C., Griffiths, T., Jurie, F., Keysers, D., Koskela, M., Laaksonen, J., Larlus, D., Leibe, B., Meng, H., Ney, H., Schiele, B., Schmid, C., Seemann, E., Shawe-Taylor, J., Storkey, A., Szedmak, S., Triggs, B., Ulusoy, I., Viitaniemi, V., & Zhang, J. (2005). The 2005 pascal visual object classes challenge. In *First PASCAL Challenges Workshop*. Springer-Verlag.

18. Exarchos, T.P., Tsipouras, M.G., Papaloukas, C., & Fotiadis, D.I. (2009). An optimized sequential pattern matching methodology for sequence classification. *Knowledge Information System* 19:249–264

19. Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., & Uthurusamy, R. (1996). Advances in Knowledge Discovery and Data Mining. *AAAI/MIT Press*

20. Flickner, M., Sawhney, H., Niblack, W., Ashley, J., Huang, Q., Dom, B., Gorkani, M., Hafner, J., Lee, D., Petkovic, D., Steele, D., & Yanker, P. (1995). Query by image and video content: The Qbic system. *IEEE Computer*, pp. 23–32

21. Gao, Y., Fan, J., Xue, X., & Jain, R. (2006). Automatic Image Annotation by Incorporating Feature Hierarchy and Boosting to Scale up SVM Classifiers. In *Proc. of ACM Int'l Conf. on Multimedia*

22. Gevers, T., & Smeulders, A. (2000). Pictoseek: Combining color and shape invariant features for image retrieval. *IEEE Trans. Image Processing*, 102–119

23. Goethals, B. (2005). Survey on Frequent Pattern Mining. Online technical report. http://www.adrem.ua.ac.be/bibrem/pubs/fpm_survey.pdf

24. Gouda, K., & Zaki, M.J. (2005). GenMax: An efficient Algorithm for mining maximal frequent itemsets. *Data Mining and Knowledge Discovery*, 11, 1-20, 2005, Springer Science and Business Media Netherlands

25. Grahne, G., & Zhu, J. (2003). High performance mining of maximal frequent itemsets. In: *Proceeding of the 6th SIAM International Workshop on High Performance Data Mining*, pp 135–143

26. Griffin, G., Holub, A., & Perona, P. (2007). Caltech-256 object category dataset. (Report No. 7896). Retrieved from California Institute of Technology Website http://authors.library.caltech.edu/7694.

27. Gunopulos, D., Khardon, R., Mannila, H., Saluja, S., Toivonen, H., & Sharma, R.S. (2003). Discovering all most specific sentences. In: ACM Transactions on Database Systems (TODS), Vol. 28, Issue 2

28. Gupta, A., & Jain, R. (1997). Visual information retrieval. *Communications of the ACM* 40, 5, 70–79

29. Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *J. Mach. Learn.* Res.

30. H'ajek P., Havel I., & Chytil M. (1966). The GUHA method of automatic hypotheses determination. Computing 1, pp 293-308

31. Han, J., Cheng, H., Xin, D., & Yan, X. (2007). Frequent pattern mining: current status and future directions. In: *Data Min Knowledge Disc.* 15:55–86

32. Han, J., Dong, G., & Yin, Y. (1999). Efficient mining of partial periodic patterns in time series database. In *Proc. 1999 Int. Conf. Data Engineering (ICDE'99)*, pages 106–115, Sydney, Australia

33. Han, J., & Pei, J. (2000). Mining frequent patterns by pattern growth: Methodology and implications. In: *SIGKDD Explorations* 2, 2, 14–20

34. Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. In: *Proceeding of the 2000 ACM-SIGMOD international conference on management of data (SIGMOD'00)*, Dallas, TX, pp 1–12

35. Holt, J.D., & Chung, S.M. (2001). Multipass algorithms for mining association rules in text databases. *Knowledge Information Systems* 3(2):168–183

36. Huang, J., et al. (1997). Image indexing using color correlogram. *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, pp 762-768, Puerto Rico

37. Jeon, J., Lavrenko, V., & Manmatha, R. (2003). Automatic Image Annotation and Retrieval using Cross-Media Relevance Models. In *Proc. of SIGIR'03*

38. Jin, R., Chai, J.Y., & Si, L. (2004). Effective Automatic Image Annotation via A Coherent Language Model and Active Learning. In *Proc. of ACM Int'l Conf. on Multimedia*

39. Kamber, M., Han, J., & Chiang, J.Y. (1997). Meta-rule-guided mining of multi-dimensional association rules using data cubes. In: *Proceeding of the 1997 international conference on knowledge discovery and data mining (KDD'97)*, Newport Beach, CA, pp 207–210

40. Keogh, E., Lonardi, S., & Ratanamahatana, C.A. (2004). Towards parameter-free data mining. In: *KDD'04*

41. Khayal, M. S. H., Rahman S., Khan D., & Salam A. (2007). Similarity based mining for finding frequent itemsets. *In: Proceedings of the 8th International conference on computers, communications, and systems,* Daegu University Korea, pp 95-100

42. Salam A., Khayal, M. S. H. (2011). Mining *top-k* frequent patterns without minimum support threshold. *Knowledge & Information Systems*, 27:112-42

43. Lai, T.S. (2000). CHROMA: a Photographic Image Retrieval System. *PhD thesis*, School of Computing, Engineering and Technology, University of Sunderland, UK http://osiris.sund.ac.uk/~cs0sla/thesis/

44. Lee, G., Lee, K.L., & Chen, A.L.P. (2001). Efficient graph-based algorithms for discovering and maintaining association rules in large database

45. Lent, B., Swami, A., & Widom, J. (1997). Clustering association rules. In *Proc. 1997 Int. Conf. Data Engineering (ICDE'97)*, pages 220–231, Birmingham, England

46. Li, J., Dong, G., & Ramamohanrarao, K. (2000). Making use of the most expressive jumping emerging patterns for classification. In: *Proceeding of the 2000 Pacific-Asia conference on knowledge discovery and data mining (PAKDD'00)*, Kyoto, Japan, pp 220–232

47. Li, J., & Wang, J.Z. (2006). Real-Time Computerized Annotation of Picture. In: *Proc. of ACM Int'l Conf. on Multimedia*

48. Li, W., Han, J., & Pei, J. (2001). CMAR: accurate and efficient classification based on multiple class-association rules. In: *Proceeding of the 2001 international conference on data mining (ICDM'01)*, San Jose, CA, pp 369–376

49. Liu, B., & Pan, J. (2007). A graph-based algorithm for mining maximal frequent itemsets. *Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007) IEEE.*

50. Liu, G., Li, J., & Wong, L. (2008). A new concise representation of frequent itemsets using generators and a positive border. *Knowledge Information System* 17:35–56

51. Liu, J., Li, M., Ma, W.Y., Liu, Q., & Lu, H. (2006). An Adaptive Graph Model for Automatic Image Annotation. *ACM Int'l Conf. on Multimedia Information Retrieval*

52. Loeff, N., Alm, C.O., & Forsyth, D.A. (2006). Discriminating image senses by clustering with multimodal features. In: *Proc. Of the COLING/ACL*, Sydney, Australia

53. Long, F., Zhang, H., & Feng, D.D. (2006). Fundamentals of content-based image retrieval. Springer Berlin.

54. Lowe, D. (2004). Distinctive image features from scale invariant keypoints. *IJCV 2(60) (91–110)*

55. Manjunath, B.S., & Ma, W.Y. (1996). Texture features for Browsing and retrieval of image data. *IEEE transactions on pattern analysis and machine intelligence*, vol, 18. No. 8

56. Mannila, H., Toivonen, H., & Verkamo, A.I. (1997). Discovery of frequent episodes in event sequences. *Data Mining Knowledge Discovery* 1(3):259–289

57. Miller, G.A., Beckwith, R., Fellbaum, C., Gross, D., & Miller, K.J. (1990). Introduction to WordNet: an on-line lexical database. *Intl. Jour. of Lexicography*, vol. 3, pp. 235-244

58. Pang-Ning, T., Vipin, K., & Jaideep, S. (2004). Selecting the right objective measure for association analysis. *Information Systems* 29:293-313

59. Pasquier, N., Bastide, Y., Taouil, R., & Lakhal, L. (1999). Discovering frequent closed itemsets for association rules. In: *Proceeding of the 7th international conference on database theory (ICDT'99)*, Jerusalem, Israel, pp 398–416

60. Pass, G., & Zabith, R. (1996). Histogram refinement for content-based image retrieval. *IEEE Workshop on Applications of Computer Vision*, pp. 96-102

61. Pei, J., Han, J., & Mao, R. (2000). CLOSET: An efficient algorithm for mining frequent closed itemsets. In: *Proceedings of the 5th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pp 21–30.

62. Piatetsky-Shapiro, G. (1991). Discovery, analysis and presentation of strong rules. In: G. Piatetsky-Shapiro, W. Frawley (Eds.), *Knowledge Discovery in Databases*, MIT Press, Cambridge, MA, pp. 229–248

63. Quelhas, P., Monay, F., Odobez, J.M., Gatica-Perez, D., Tuytelaars, T., & Gool, L.V. (2005). Modeling scenes with local descriptors and latent aspects. In: *IEEE Intl. Conf. on Computer Vision*

64. Ramaswamy, S., Mahajan, S., & Sillberschatz, A. (1998). On the discovery of interesting patterns in association rules. In: *Proceeding of Int. Conf. Very large Databases (VLDB' 98)*, New York, pp 368-379

65. Rege, M., Dong, M., & Fotouhi, F. (2007). Building a user-centered semantic hierarchy in image databases

66. Russell, B.C., Torralba, A., Murphy, K.P., & Freeman, W.T. (2008). LabelMe: a database and web-based tool for image annotation. *International Journal Of Computer Vision*, Volume 77, Issue 1-3, Pages 157-173

67. Russ, J.C. (1998). The Image Processing Handbook. CRC Press, *IEEE Press*, third edition

68. Saux, B.L., & Amato, G. (2004). Image recognition for digital libraries. *ACM MIR'04*

69. Silverstein, C., Brin, S., Motwani, R., & Ullman, J.D. (1998). Scalable techniques for mining causal structures. In: *Proceeding of the 1998 international conference on very large data bases (VLDB'98)*, New York, NY, pp 594–605

70. Sivic, J., Russell, B., Efros, A., Zisserman, A., & Freeman, W. (2005). Discovering objects and their location in images. In *IEEE Intl. Conf. on Computer Vision*

71. Smeulders, A.W.M., Worring, M., Santini, S., Gupta, A., & Jain, R. (2000). Content-based image retrieval at the end of early years. *IEEE Trans. Patern Anal. Mach. Intell*, 22, 1349-1380

72. Smith, J.R., & Chang, S.F. (1996). Visualseek: a fully automated content-based image query system. In: *ACM International Conference on Multimedia*, pages 87–98, Boston, MA

73. Srikanth, M., Varner, J., Bowden, M., & Moldovan, D. (2005). Exploiting Ontologies for Automatic Image Annotation. In *Proc. of ACM SIGIR'05*

74. Sudderth, E., Torralba, A., Freeman, W.T., & Willsky, W. (2005). Describing visual scenes using transformed dirichlet processes. In *Advances in Neural Info. Proc. Systems*

75. Tan, P-N., & Kumar, V. (2000). Interestingness measures for association patterns: a perspective, In: *KDD 2000 Workshop on Postprocessing in Machine Learning and Data Mining*, Boston, MA

76. Tianming, H., Sam, Y.S., Hui, X., & Qian, F. (2007). Discovery of maximum length frequent itemsets.

77. Tsai, C.F., & Hung, C. (2008). Automatically annotating images with keywords: A review of image annotation systems. *Recent Patents on Computer Science*, 55-68

78. Van Rijsbergen, C.J. (1979). Information Retrieval, 2nd Edition, Butterworths, London

79. Wang, H., Yang, J., Wang, W., & Yu, P. (2002). Clustering by pattern similarity in large data sets. In: *Proceedings of the ACM SIGMOD International conference on management of data*, Wisconsin, pp 394–405

80. Wang, J., Han, J., & Pei, J. (2003). CLOSET+: searching for the best strategies for mining frequent closed itemsets. In: *Proceeding of the ACM SIGKDD International conference on knowledge discovery and data mining (KDD'03)*, Washington, DC, pp 236–245

81. Wang, J., Han, J., Lu, Y., & Tzvetkov, P. (2005). TFP: an efficient algorithm for mining top-k frequent closed itemsets. In: *TKDE*

82. Wang, J., & Karypis, G. (2005). HARMONY: efficiently mining the best rules for classification. In: *Proceeding of the 2005SIAMconference on data mining (SDM'05)*, Newport Beach, CA, pp 205–216

83. Wang, J., Li, J., & Wiederhold, G. (2001). Simplicity: Semantics-sensitive integrated matching for picture libraries. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 947–963

84. Wang, J.Z., Boujemaa, N., Del Bimbo, A., Geman, D., Hauptmann, A., & Tesic, J. (2006). Diversity in multimedia information retrieval research. In *Proc. MIR Workshop, ACM Multimedia*

85. Wu, X., Zhang, C., & Zhang, S. (2004). Efficient mining of both positive and negative association rules. *ACM Trans Information System* 22(3): 381–405

86. Yang, G. (2004). The complexity of mining maximal frequent itemsets and maximal frequent patterns. In: *Proceeding of the ACM SIGKDD International conference on knowledge discovery in databases (KDD'04)*, Seattle, WA, pp 344–353

87. Yavlinsky, A., Schofield, E., & Ruger, S. (2005). Annotation Using Global Features and Robust Nonparametric Density Estimation. In: *Proc. of Int'l Conf. on Image and Video Retrieval*

88. Yen, S.J., & Chen, A.L.P. (1996). An efficient approach to knowledge discovery from large databases. In *Proceedings of the IEEE/ACM International Conference on Parallel and Distributed Information Systems*. ACM Press, 8–18.

89. Yilmaz, A., Javed, O., & Shah, M. (2006). Object tracking: A Survey. *ACM Computing Surveys*, Vol. 38, No. 4, Article 13

90. Yin, X., & Han, J. (2003). CPAR: classification based on predictive association rules. In: *Proceeding of the 2003 SIAM international conference on data mining (SDM'03)*, San Fransisco, CA, pp 331–335

91. Zaki, M.J., & Hsiao, C.J. (2002). CHARM: An efficient algorithm for closed itemset mining. In: *Proceedings of the 2nd SIAM International Conference on Data Mining*, pp 12–28

92. Zhang, R., Zhang, Z., & Khanzode, S. (2004). A data mining approach to modeling relationships among categories in image collection. *ACM KDD'04*

93. Zhang, S., Wu, X., Zhang, C., & Lu, J. (2008). Computing the minimum-support for mining frequent patterns. *Knowledge Information System* 15:233–257

Appendices

Appendix-A: Paper-I

SIMILARITY BASED MINING FOR FINDING FREQUENT ITEMSETS

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

Abdus Salam
INTERNATIONAL ISLAMIC UNIVERSITY ISLAMABAD (IIUI) PAKISTAN
abdusslam@hotmail.com

Saif-ur-Rahman
INTERNATIONAL ISLAMIC UNIVERSITY ISLAMABAD (IIUI) PAKISTAN
saifeyabbas@yahoo.com

Dawlat Khan
INTERNATIONAL ISLAMIC UNIVERSITY ISLAMABAD (IIUI) PAKISTAN
dawlat_khan2000@yahoo.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 $I = \{i_1, i_2, i_3, \ldots, i_m\}$ 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 $D = \{t_1, t_2, \ldots, t_m\}$ be the transaction database where every transaction in D has unique transaction identifier in Y. Let X is itemset if $X \subseteq I$ and X is frequent itemset we use the following similarity measure to find FIS;

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

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 extracts 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 ~ 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.

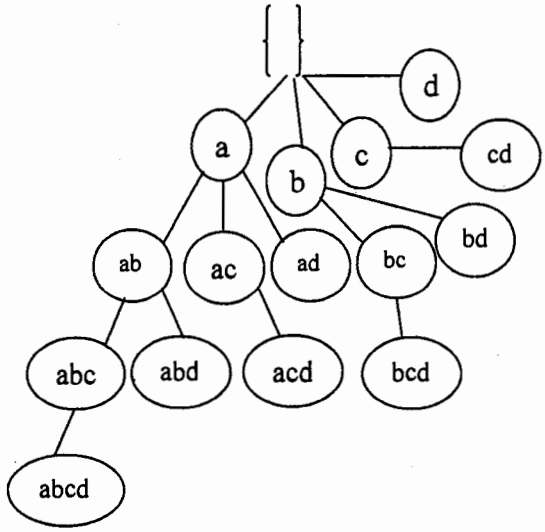


Figure 1 SE-Tree

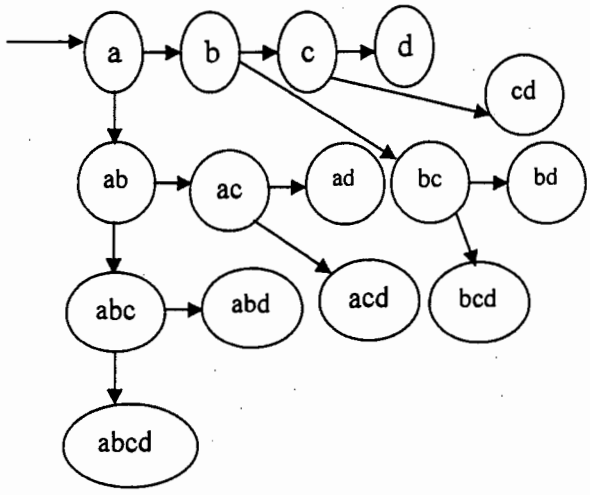


Figure 2 Alternative representation of SE-Tree

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	
TID	ITEMS
1	A, B, E
2	B, D
3	B, C
4	A, B, D
5	A, C
6	B, C
7	A, C
8	A, B, C, E
9	A, B, C

HORIZENTAL ITEM SET					
1	A	B	E		
2	B	D			
3	B	C			
4	A	B	D		
5	A	C			
6	B	C			
7	A	C			
8	A	B	C	E	
9	A	B	C		

VERTICAL TID SET				
A	B	C	D	E
1	1	3	2	1
4	2	5	4	8
5	3	6		
7	4	7		
8	6	8		
9	8	9		
	9			

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, **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 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

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

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

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

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

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

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

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, $A_1, A_2, \ldots, A_n$, the number of the elements in union $A_1 \cup A_2 \cup \ldots \cup A_n$ 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.

$$\left| A_1 \bigcup A_2 \bigcup \ldots\ldots A_n \right| = \sum_i |A_i| - \sum_{i<j} \left| A_i \bigcap A_j \right| + \sum_{i<j<k} \left| A_i \bigcap A_j \bigcap A_k \right| - \ldots + (-1)^{n+1} \left| A_1 \bigcap A_2 \bigcap \ldots\ldots \bigcap A_n \right| \quad (2)$$

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 currents 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
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. We have noticed that lesser number of FIS are generated as the size of FIS is increased and vice versa.

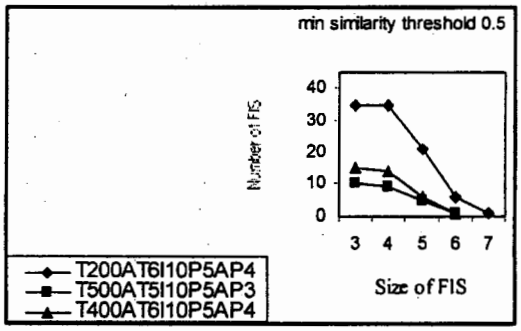


Figure 5 FIS generation with SB-Miner

In order to verify the correctness of our technique we have applied apriori and FPgrowth 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.*

Appendix-B: Paper-II

Acceptance Letter

Date: Oct 16, 2010
To: "Abdus Salam" abdusslam@hotmail.com
cc: kais@cems.uvm.edu, weiliwu@utdallas.edu
From: "Xindong Wu" kais@cems.uvm.edu
Subject: KAIS Regular Paper KAIS-2030R4: Acceptance and Final Manuscript

Dear Mr. Salam,

"Mining top-k frequent patterns without minimum support threshold"
by Abdus Salam, MS; Sikandar H Khiyal, PhD

Received: Apr 26, 2009
Revised: Oct 07, 2010
Accepted: Oct 16, 2010

(Note: These dates should be provided in the final manuscript, and the author(s) are not allowed to change them.)

I am pleased to inform you that the above revised paper has been accepted as a Regular Paper for publication in the Knowledge and Information Systems (KAIS) journal. Please try and update the older KAIS papers in the references with 2009 or 2010 published papers from the journal.

Please follow the instructions for final manuscript preparation on the journal home page (http://www.cs.uvm.edu/~kais/). Please do not make any deletions to your article and submit the following items *within 5 weeks*.

* Electronic submission of your final manuscript

Please send an electronic copy of the final manuscript to srilakshmi Patrudu at Srilakshmi.Patrudu@springer.com

Editorial Manuscript Number: KAIS-2030R4

You should (a) include all source files and a PDF (or PS) file of the final paper, (b) use a zip archive or any other archiving software (tar, tar.gz, tgz, tar.Z, taz, or gz), and (c) submit the zip archive as a single file. After uploading the single file an acknowledgement of receipt will be sent to you.

(If you don't get the acknowledgement of receipt within 48 hours, please e-mail Srilakshmi.Patrudu@springer.com to check your file's arrival.)

COLOR FIGURES: The journal is printed in black and white by default. If you have color figures in your paper, you should either make sure that they are readable when

printed in black and white, or ask for
color reproduction when submitting your final manuscript by indicating that you are agreeing to pay the charges for color in the final version (see http://www.springeronline.com/sgw/cda/frontpage/0,11855,4-152-70-1136715-detailsPage%253DcontentItemPage%2526contentItemId%253D148346%2526CIPageCounter%253DCI_FOR_AUTHORS_AND_EDITORS_PAGE6,00.html#anchor6 for details).

If you cannot provide LaTeX files for your final manuscript submission, Springer also accepts Microsoft Word documents with no typesetting charge.

* For Regular Papers Only: A passport-size photo of each author

If you have included photos in the above zip-archive submission, and the photos are in the provided PDF (or PS) file, Springer does not need a hardcopy of your photos.

I congratulate you on acceptance of your manuscript for publication as a regular paper and look forward to receiving the above mentioned publication material.

If your library has not subscribed to the KAIS journal, Springer and I would strongly encourage you to contact your librarian (or other relevant people) for a subscription. The subscription form is available on the journal home page (at http://www.cs.uvm.edu/~kais/). Please note that a subscription to the journal automatically includes online access to the electronic version of the journal at http://link.springer.de/link/service/journals/10115/index.htm, in addition to the printed version.

Sincerely,

Xindong Wu
Editor-in-Chief, Knowledge and Information Systems

Knowl Inf Syst
DOI 10.1007/s10115-010-0363-3

REGULAR PAPER

Author Proof

Mining *top-k* frequent patterns without minimum support threshold

Abdus Salam · M. Sikandar Hayat Khayal

Received: 26 April 2009 / Revised: 7 October 2010 / Accepted: 4 November 2010

Abstract Finding frequent patterns play an important role in mining association rules, sequences, episodes, Web log mining and many other interesting relationships among data. Frequent pattern mining methods often produce a huge number of frequent itemsets that is not feasible for effective usage. The number of highly correlated patterns is usually very small and may even be one. Most of the existing frequent pattern mining techniques often require the setting of many input parameters and may involve multiple passes over the database. Minimum support is the widely used parameter in frequent pattern mining to discover statistically significant patterns. Specifying appropriate minimum support is a challenging task for a data analyst as the choice of minimum support value is somewhat arbitrary. Generally, it is required to repeatedly execute an algorithm, heuristically tuning the value of minimum support over a wide range, until the desired result is obtained, certainly, a very time-consuming process. Setting up an inappropriate minimum support may also cause an algorithm to fail in finding the true patterns. We present a novel method to efficiently retrieve top few maximal frequent patterns in order of significance without use of the minimum support parameter. Instead, we are only required to specify a more human understandable parameter, namely the desired number itemsets k. Our technique requires only a single pass over the database and generation of length two itemsets. The *association ratio graph* is proposed as a compact structure containing concise information, which is created in time quadratic to the size of the database. Algorithms are described for using this graph structure to discover *top-most* and *top-k* maximal frequent itemsets without minimum support threshold. To effectively achieve this, the method employs construction of an all path source-to-destination tree to discover all maximal cycles in the graph. The results can be ranked in decreasing order of significance. Results are presented demonstrating the performance advantages to be gained from the use of this approach.

A. Salam (✉)
Department of Computer Science, International Islamic University, Islamabad 10, Pakistan
e-mail: abdusslam@hotmail.com

M. S. H. Khayal
Department of Computer Science and Software Engineering,
Fatima Jinnah Women University, Rawalpindi, Pakistan
e-mail: m.sikandarhayat@yahoo.com

Journal: 10115 Article No.: 0363 MS Code: KAIS-2030.4 ☐TYPESET ☐DISK ☐LE ☐CP Disp.:2010/11/17 Pages: 30 Layout: Small

Keywords Frequent pattern mining · Maximal frequent itemsets · Association ratio · Association ratio graph · All-path-source-destination tree

1 Introduction

Frequent patterns are frequent itemsets, subsequences, sub-graphs, substructures, phrase-sets and term-sets that appear in real-world databases with a frequency no less than a specified *minimum support* (called *min_sup* in this text) threshold. Mining frequent patterns is a corner stone of various data mining tasks and applications such as associations [3], correlations [8], sequences [4], sequence classification [14], episodes [35], negative rules [49], classification [33], and clustering [46]. Frequent pattern mining is also essential to a wide range of emerging applications, such as Web log mining [15], click-stream mining [11], network traffic analysis [19], stock market analysis [31], and sensor networks [45]. With over a decade of extensive research, frequent pattern mining has become an important data mining application and a focused theme of research.

Agrawal et al. [2] proposed the frequent pattern mining by coining the term *association rules* for the market basket data analysis. In this analysis, the buying behavior of customers is analyzed by finding associations between different items that customers purchase together. However, the term association rules is understood in more general way as representation of knowledge about a relation between two Boolean attributes (created e.g. as conjunctions attribute value pairs) that are supported by the analyzed data. This comprehensive understanding of association rules, first proposed in [24], is the basis of an exploratory data analysis method known as General Unary Hypotheses Automaton (GUHA) that goes back to 1966. It is remained unnoticed for long that what is now called "association rules" in data mining has already been proposed in the paper on GUHA.

Association rule mining is a two-step process: (i) the identification of *frequent itemsets* within the dataset and (ii) the derivation of *inferences* from these itemsets. Finding frequent itemsets is recognized as the computationally most challenging task and has shown to be *NP-Complete* [22]. Therefore, majority of the methods have focused upon efficient discovery of frequent patterns as its level of complexity is significantly higher than that of inference generation. The problem arises because the number of possible itemsets is exponential in the number of possible attribute-values. Given n distinct items within a database there are 2^n possible combinations of itemsets to explore.

Most of the classic frequent pattern mining approaches [1,3,5,10,19,26] attempt to discover all distinct valid frequent itemsets with frequency no less than a *min_sup* value. Two main dangers of working with support-based algorithms are (i) setting up an incorrect *min_sup* may cause an algorithm to fail in finding the correlated patterns and (ii) a perhaps more sinister problem is that the algorithm may report spurious patterns that do not really exist. This is especially likely when an analyst fails to understand the role of an input parameter in the data mining process or may not select efficient parameter values causing an algorithm to fail in finding the highly correlated patterns [32]. These concerns are the motivating factors for continuing research to find efficient algorithms for mining frequent patterns without a preset *min_sup* threshold. Data mining algorithms should have as few parameters as possible, ideally none. A parameter-free algorithm will be free of an individual's prejudices and presumptions on the problem and allow the data to speak for itself. While specifying the mining requirements for frequent itemsets, the term 'frequent' is already a threshold from a fuzzy viewpoint [52]. However, existing algorithms still require analysts to specify the appropriate *min_sup* to mine databases, a challenging task for an analyst with little domain

Journal: 10115 Article No.: 0363 MS Code: KAIS-2030.4 TYPESET DISK LE CP Disp.:2010/11/17 Pages: 30 Layout: Small

knowledge. Even when *min_sup* is adjusted under the supervision of an experienced miner, it cannot be ascertained whether the results are what is desired.

The bottleneck of frequent pattern mining does not arise when we derive the *complete* set of frequent patterns under certain constraints rather it arises when we want to extract a *compact but high quality* set of patterns that are most useful in applications [24]. Majority of existing frequent pattern mining methods often produce a huge number of frequent itemsets that is not feasible for effective usage. There are proposals on reduction in large result sets, such as closed patterns, maximal patterns, approximate patterns, condensed pattern bases, representative patterns, clustered patterns, and discriminative frequent patterns. However, it is still not clear what kinds of patterns give satisfactory results that are compact and carry representative quality for a particular application, and if we can mine such patterns without *min_sup* threshold.

In this paper, we show that it is possible to discover a small set of highly correlated frequent itemsets without predetermined *min_sup* value to enable the better feasibility of mining compact but high-quality frequent patterns. Moreover, in very large databases the number of highly correlated patterns is usually very small and may even be *one*. Thus, the quality of retained patterns can be enhanced substantially by reducing the size of derived pattern sets. Experimental results show that our approach offers significantly better performance than existing support-based methods. We present two novel algorithms for mining *top-most* and *top-k* patterns without *min_sup* using graph-theoretic approach.

The contribution of this paper can be summarized as follows: (i) most previous works focus on improving the search time, whereas we concentrate on mining without a preset *min_sup* threshold. (ii) While previous works generate length $1-$itemsets, $2-$itemsets,..., $n-$itemsets, we show that generating only length 2-itemsets is sufficient to discover desired patterns. (iii) The performance of most of the methods suffers considerably due to multiple database scans. We have shown that only single database pass produces the desired results and can achieve tremendous improvement in terms of speed and efficiency. (iv) We propose an association measure that converts the co-occurrence frequency counts of database items into proximity measure that nullifies the least associated itemsets.

This paper is organized as follows. Section 2 introduces the formal problem description and set up basic definitions. In Sect. 3 we review the related work. In Sect. 4, we give the design of *top-most* and *top-k* maximal frequent pattern generation algorithms. The implementation details and experimental results are presented in Sect. 5. Finally, this paper concludes with Sect .6.

2 Problem description

A formal specification of the problem is presented as follows: Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of m distinct items. A database $D = \{t_1, t_2, \ldots, t_n\}$ consists of a set of n transactions, where each transaction t_i is a subset of I or $t_i \subseteq I$. Each transaction has a unique identifier (tid). A set $X \subseteq I$ is called an *itemset* (items in an itemset are distinct and independent of the order), and $|X|$ represents the number of items in an itemset. Itemsets of length k are referred to as *k-itemsets*. The support of an itemset X, denoted by $supp(X)$, is the fraction of transactions in D containing X. An itemset X is a frequent itemset if the $supp(X)$ is larger than a *min_sup* value, indicating that the presence of itemset X is significant in the database. Given a *min_sup* value, the goal of frequent pattern mining is to efficiently enumerate all frequent patterns that satisfy the input constraint.

Journal: 10115 Article No.: 0363 MS Code: KAIS-2030.4 TYPESET DISK LE CP Disp.:2010/11/17 Pages: 30 Layout: Small

2.1 Mining frequent itemsets

Frequent itemsets are patterns that appear in a dataset with frequency no less than a specified threshold. The task of discovering frequent itemsets is quite challenging as the search space is exponential with respect to the number of items occurring in the database. The minimum support threshold parameter is used to limit the output to a hopefully reasonable size. Necessary definitions for compressing the frequent pattern result sets are as follow:

Definition 1 (*frequent itemsets* (***FI***)) Given the minimum support threshold *min_sup*, an itemset X is called frequent itemset where $supp(X)$ is greater than *min_sup* i.e., $F_I = \{X | supp(X) \geq min\_sup\}$.

Setting up an appropriate *min_sup* is usually left unresolved to users in previous works. A major challenge arises when the value of *min_sup* is set low. A small *min_sup* value may generate an exponential number of frequent patterns since such a small value only prunes a few itemsets. As the number of derived frequent itemsets is very large, it is not practical to produce and use such a huge result set. It is required to compress the number of derived patterns to a considerable limit to enhance the quality of desired patterns. There have been concerted efforts to substantially reduce the huge set of derived patterns while maintaining the high quality of patterns by focusing on mining a compressed or approximate set of frequent patterns.

Pasquier et al. [38] focused on mining a smaller set of frequent patterns called closed frequent itemset ((CFI) as explained in Definition 2 below) to compress the patterns in terms of the information that the result set contains. Presenting an Apriori-based algorithm called A-Close [38] to reduce the size of derived pattern sets that still contain enough information to generate association rules. Other closed pattern mining algorithms include CLOSET [39], CHARM [51] and CLOSET+ [47]. Mining closed frequent itemsets provides an interesting and important alternative to mining frequent itemsets since it inherits the same analytical power but generates a much smaller set of results.

Definition 2 (*closed frequent itemsets* (***CFI***)) A pattern α is a *closed frequent itemset* in a dataset D if α is frequent in D and there exists no proper superset β such that β has the same support as α in D [24].

In proposals for further reducing the size of discovered itemsets, Bayardo [5] developed an Apriori-based method (MAX-Miner) that calls for generation of *maximal frequent itemset* for search space reduction. Yang [50] provides theoretical analysis of the (worst-case) complexity of mining maximal patterns.

Definition 3 (*maximal frequent itemsets* (***MFI***)) A pattern Φ is a *maximal frequent itemset* in dataset D if Φ is frequent, and there exists no superset Φ such that $\subset$ and Φ is frequent in D [24].

Tianming Hu et al. [43] propose mining maximum length frequent itemset to further reduce the size of derived pattern sets.

Definition 4 (*maximum length frequent itemset* (***MLFI***)) An itemset X is a maximum length frequent itemset if $supp(X) \geq min\_sup$ and for all itemset Y, if $supp(Y) \geq min\_sup$ then $|X| \geq |Y|$, where $|Y|$ and $|X|$ denote the number of items contained in Y and X respectively [43].

Journal: 10115 Article No.: 0363 MS Code: KAIS-2030.4 TYPESET DISK LE CP Disp.:2010/11/17 Pages: 30 Layout: Small

Table 1 An example transaction database

Tid	Items
1	A B D F
2	A B P
3	A D F
4	B C D E
5	B C D E F
6	A B F
7	A B D F
8	A B F
9	A B C D F
10	A B C E F

The relationship between the set of all frequent itemsets (FIs), the set of closed frequent itemsets (CFIs), and the set of maximal frequent itemsets (MFIs) is {MFIs} $\subseteq$ {CFIs} $\subseteq$ {FIs} [14]. The set MFI is typically orders of magnitude smaller than the set CFI, which itself is orders of magnitude smaller than the set FI. Thus, it is impractical to generate the entire set of frequent itemsets or closed frequent itemsets when the length of patterns is very long (patterns containing many items) in the database. Although numerous scalable methods have been developed to compress the resulting patterns, such methods often generate a huge number of frequent patterns. In many dense databases, the set of CFIs could be still too large and orders of magnitude larger than the set of MFIs [10,19]. What are interesting patterns and how to mine them efficiently? To answer this question, the only recourse is to mine top few maximal frequent itemsets.

2.2 The role of minimum support

Setting up an appropriate *min_sup* is quite subtle. Choosing a small *min_sup* can result in an extremely large size of frequent itemsets at the cost of execution efficiency. Contrary to this setting, a large *min_sup* may only generate few but insignificant itemsets. There is a need to tune the *min_sup* over a wide range to discover highly correlated patterns. This is very time-consuming and indeed a serious problem for the applicability of mining frequent itemsets. Since our focus is to discover top few maximal frequent itemsets without *min_sup* threshold, it is important to show how the size of MFIs changes with respect to different *min_sup* thresholds. We will illustrate this by the example below.

Consider the transaction database of Table 1. The database contains six different items, I = {A, B, C, D, E, F} and ten transactions. We illustrate the generation of FI, CFI and MFIs at four different *min_sup* values using the Apriori algorithm. The results are shown in Fig. 1, where Fig. 1a, b,c, and d show derived itmsets at 20, 40, 60, and 80% *min_sup* values respectively. From the example shown in Fig. 1, we can clearly see that the number of FIs, CFIs, and MFIs changes with respect to different *min_sup* thresholds. If x and y are two different *min_sup* values and if $x > y$ then $FI_x(D) \subseteq FI_y(D)$ i.e., the number of frequent itemsets decreases with increase in support thresholds. However, this anti-monotonicity property of FIs does not hold for MFIs i.e., $|MFI_{20}(D)| = 5, |MFI_{40}(D)| = 4, |MFI_{60}(D)| = 2$, and $|MFI_{80}(D)| = 2$. From the below example, we can observe that MFIs behave rather randomly, and this adds more difficulty to discover MFIs without *min_sup* threshold.

Journal: 10115 Article No.: 0363 MS Code: KAIS-2030.4 TYPESET DISK LE CP Disp.:2010/11/17 Pages: 30 Layout: Small

At 20% Minimum Support

Itemsets	Support Count	FI	FCI	MFI
A	8	Yes	No	No
B	9	Yes	Yes	No
C	4	Yes	No	No
D	6	Yes	Yes	No
E	3	Yes	No	No
F	9	Yes	Yes	No
A B	7	Yes	No	No
A C	2	Yes	No	No
A D	4	Yes	No	No
A F	8	Yes	Yes	No
B C	4	Yes	Yes	No
B D	5	Yes	Yes	No
B E	3	Yes	No	No
B F	8	Yes	Yes	No
C D	3	Yes	No	No
C E	3	Yes	No	No
C F	3	Yes	Yes	No
D E	2	Yes	No	No
D F	5	Yes	Yes	No
E F	2	Yes	No	No
A B C	2	Yes	No	No
A B D	3	Yes	No	No
A B F	7	Yes	Yes	No
A C F	2	Yes	No	No
A D F	4	Yes	Yes	No
B C D	3	Yes	Yes	No
B C E	3	Yes	Yes	No
B C F	3	Yes	Yes	No
B D E	2	Yes	No	No
B D F	4	Yes	Yes	No
B E F	2	Yes	No	No
C D E	2	Yes	No	No
C D F	2	Yes	No	No
C E F	2	Yes	No	No
A B C F	2	Yes	Yes	Yes
A B D F	3	Yes	Yes	Yes
B C D E	2	Yes	Yes	Yes
B C D F	2	Yes	Yes	Yes
B C E F	2	Yes	Yes	Yes
Total		39	20	5

(a)

At 40% Minimum Support

Itemsets	Support Count	FI	FCI	MFI
A	8	Yes	No	No
B	9	Yes	Yes	No
C	4	Yes	No	No
D	6	Yes	Yes	No
F	9	Yes	Yes	No
A B	7	Yes	No	No
A D	4	Yes	No	No
A F	8	Yes	Yes	No
B C	4	Yes	Yes	Yes
B D	5	Yes	Yes	No
B F	8	Yes	Yes	No
D F	5	Yes	Yes	No
A B F	7	Yes	Yes	Yes
A D F	4	Yes	Yes	Yes
B D F	4	Yes	Yes	Yes
Total		15	11	4

(b)

At 60% Minimum Support

Itemsets	Support Count	FI	FCI	MFI
A	8	Yes	No	No
B	9	Yes	Yes	No
D	6	Yes	Yes	Yes
F	9	Yes	Yes	No
A B	7	Yes	No	No
A F	8	Yes	Yes	No
B F	8	Yes	Yes	No
A B F	7	Yes	Yes	Yes
Total		8	6	2

(c)

At 80% Minimum Support

Itemsets	Support Count	FI	FCI	MFI
A	8	Yes	No	No
B	9	Yes	Yes	No
F	9	Yes	Yes	No
A F	8	Yes	Yes	Yes
B F	8	Yes	Yes	Yes
Total		5	4	2

(d)

Fig. 1 The effect of varying minimum support threshold

Moreover, the itemset {A, B, F} can best illustrate the inconsistency of support-based approach. At 20% support {A, B, F} is frequent and closed but not maximal, whereas, at 40 and 60% support it also becomes maximal. It is also evident in Fig. 1 that a closed itemset is a frequent itemset, and a maximal itemset is also a closed itemset and $|MFI| < |CFP| < |FI|$. This establishes that we have to tune the *min_sup* over a wide range to obtain desired patterns.

Mining frequent patterns in a large search space makes full use of the pruning power of the *min_sup* to cut down the search space as well as the result set. All the compression methods discussed in the previous section such as FI, CFI, MFI, and MLFI rely on *min_sup* threshold. Eliminating *min_sup* from the mining process requires some heuristic strategies to reduce

Journal: 10115 Article No.: 0363 MS Code: KAIS-2030.4 TYPESET DISK LE CP Disp.:2010/11/17 Pages: 30 Layout: Small

Table 2 Association measures and their formulae

#	Measure	Formula
1	Interest (I)	$\frac{P(A,B)}{P(A)P(B)}$
2	Cosin (IS)	$\frac{P(A,B)}{\sqrt{P(A)P(B)}}$
3	Jacaard (ς)	$\frac{P(A,B)}{P(A)+P(B)-P(A,B)}$
4	Piatetsky-Shapiro (PS)	$P(A,B) - P(A)P(B)$

the search space and the output size. Our intuition is based on to bring frequent items close to each other using a similarity measure also called association measure in such a way to form clusters of frequent patterns. Once frequent pattern clusters are ready, it is easy to find top few maximal frequent itemsets in top–down fashion.

2.3 Association measure

The selection of right similarity measure is imperative to capture association among items. Many similarity measures are reported in the literature to compute the proximity between the independent variables. For more information about different association measures, readers are referred to Pang-Ning et al. [36]. However, many such measures provide conflicting information about the interestingness of a pattern, and the best measure to use for a given application domain is hardly known. Their study in [36] shows that there is no measure that is consistently better than others in all application domains. This is because each association measure has its own diversified characteristics and is designed for specific problems. Several well-known symmetric association measures have been used extensively to evaluate the interestingness of association patterns—the stronger is the dependence relationship, the more interesting is the pattern. These measures, interest factor (I) [9], cosine (IS) [42], Jaccard (ζ) [44] and Piatetsky-Shapiro (PS) [40], are defined in terms of probabilities of relative frequency counts, as shown in Table 2.

We compute association between 2-itemsets on the example database given in the Table 1 using the association measures specified in the Table 2 to indicate their conflicts. The resulting values are ranked according to their association measure in decreasing order of magnitude as shown in Table 3. The rankings are not identical for all the measures. Table 3 shows that different measures can lead to substantially different rankings, a significant number of them provide conflicting information about the interestingness of a pattern. For instance, the itemset {C, E} is ranked highest by the interest (I) and Piatetsky-Shapiro (PS) measures but ranked third according to cosine (IS) and Jaccard (ζ) measures. The itemsets {A, F} and {B, F} having the highest support count as shown in Fig. 1a are the most frequent patterns but interest (I) and Piatetsky-Shapiro (PS) do not rank these itemsets highest in the order in Table 3.

Consider itemsets {C, D}, {C, E} and {C, F}. As can be seen item C is common in all three itemsets. The individual item support count for C and E being the lowest and 2-itemset support count of all the three itemsets is also low i.e., 3 as shown in Fig. 1a. These itemsets should be ranked lowest in the order due to a small support count value. No association measure produces the proper ordering of the itemsets. The itemsets ranking is proper when most frequent itemsets are ranked at the top and least frequent itemsets are ranked in the bottom of the ranking list. These conflicting results render these measures inappropriate to be used for computation of association among database items. Thus, there arises a need for

Journal: 10115 Article No.: 0363 MS Code: KAIS-2030.4 TYPESET DISK LE CP Disp.:2010/11/17 Pages: 30 Layout: Small

Table 3 Conflicts in ranking of 2-itemsets by different association measures

Interest (I)		Cosine (CS)		Jaccard (ç)		Piatetsky_Shapiro (PS)	
Itemsets	Association	Itemsets	Association	Itemsets	Association	Itemsets	Association
C E	0.250	A F	0.943	A F	0.889	C E	0.180
C D	0.125	B F	0.889	B F	0.800	A F	0.080
A F	0.111	C E	0.866	C E	0.750	C D	0.060
B C	0.111	A B	0.825	C F	0.750	B C	0.040
B E	0.111	B D	0.680	A B	0.700	B E	0.030
D E	0.111	D F	0.680	B D	0.500	D E	0.020
B F	0.099	B C	0.667	D F	0.500	B F	−0.010
A B	0.097	C D	0.612	B C	0.444	A B	−0.020
B D	0.093	A D	0.577	C D	0.429	B D	−0.040
D F	0.093	B E	0.577	A D	0.400	D F	−0.040
A D	0.083	C F	0.500	B E	0.333	C F	−0.060
C F	0.083	D E	0.471	D E	0.286	E F	−0.070
E F	0.074	E F	0.385	A C	0.200	A D	−0.080
A C	0.063	A C	0.354	E F	0.200	A C	−0.120
A E	0.042	A E	0.204	A E	0.100	A E	−0.140

an appropriate association measure (Sect. 4.1) which can lead to proper ordering of itemsets considering the intrinsic properties of relative frequencies.

3 Related work

The Apriori [3] is the first algorithm developed for finding frequent itemsets and many of the proposed algorithms are its variants. Apriori employs a bottom–up, breadth-first search that enumerates every single frequent itemset and performs well when the size of the MFIs is small but requires too much time if the maximal itemset size is large. Many algorithms, such as direct hashing and pruning (DHP) [37], Partition [41], DIC [8], Multipass [28] and inverted hashing and pruning (IHP) [29], are based on Apriori and use various approaches to improve candidate generation and pruning. The efficiency of Apriori-like algorithms is low due to the exponential enumeration of candidate itemsets and repeated database scans at each level for support checking.

Max-Miner [5] uses set enumeration tree to enumerate all FIs. It employs a breadth-first search, which reduces database scans by employing a look-ahead pruning strategy. It reduces the search space by pruning the tree to eliminate both supersets of infrequent itemsets and subsets of frequent itemsets. The Max-Miner computes the support of an itemset, it also computes the support of the largest itemset that appears in the subtree rooted at this itemset. If this superset is frequent, then all other itemsets in this subtree must be frequent too. Thus, the algorithm avoids having to compute the support of all the frequent itemsets. Since Max-Miner uses the original horizontal database format, it can perform the same number of passes over a database as Apriori does.

FP-growth [26] is a fundamentally different algorithm from the Apriori-like algorithms. FP-growth finds FIs without candidate set generation and converts the database into a compact

Journal: 10115 Article No.: 0363 MS Code: KAIS-2030.4 TYPESET DISK LE CP Disp.:2010/11/17 Pages: 30 Layout: Small

FP-tree structure to avoid repeated database scans. It achieves great performance gains against Apriori-like algorithms due to compressed representation of the entire database in the main memory. However, it is not scalable to very large databases, due to the main memory requirements of the FP-trees. Since it enumerates all frequent patterns, it is impractical when pattern length is long.

DepthProject [1] discovers long itemsets using a depth-first search of a lexicographic tree of itemsets and uses a counting method based on transaction projections along its branches. This projection is equivalent to a horizontal version of the *tid* sets at a given node in the search tree. The DepthProject also uses look-ahead pruning method with item reordering. It returns a superset of the MFI and would require post-pruning to eliminate non-maximal patterns.

MAFIA [10] is one of the state-of-the-art algorithms for finding MFIs. Mafia uses three pruning strategies to remove non-maximal sets; (i) the look-ahead pruning strategy first used in MaxMiner, (ii) checks if a new set is subsumed by an existing maximal set, and (iii) the last technique checks if $t(X) \subset t(Y)$, if so X is considered together with Y for extension. Mafia uses vertical bit-vector data format, and compression and projection of bitmaps to improve performance. Mafia mines a superset of the MFI and requires a post-pruning step to eliminate non-maximal patterns. The most time-consuming step involve is the conversion of database into vertical bit-vectors.

GenMax [21] uses backtracking search based technique for finding MFIs. GenMax uses a number of optimizations to prune the search space. It uses *progressive focusing* to perform maximality checking, and *diffset propagation* to perform fast frequency computation. Systematic experimental comparison with previous work indicates that different methods have varying strengths and weaknesses based on dataset characteristics. It concludes that while Mafia is good for mining a *superset* of all maximal patterns, GenMax is the method of choice for enumerating the *exact* set of maximal patterns.

Zhang et al. [52] propose a new computational strategy for addressing the issue of setting up the *min_sup* threshold. Their mining strategy is different from existing Apriori-like algorithms because it allows specifying the requirements in commonly used modes. The approach uses polynomial approximation for a specified *min_sup* on the commonly used interval [0, 1] and fuzzy estimation for a specified *min_sup* in fuzzy sets. More specifically, a *min_sup* threshold is taken as input and the algorithms (using polynomial approximation and fuzzy estimation) computationally convert the specified threshold into an actual *min_sup* (appropriate to a database to be mined). This allows analysts to specify their mining requirements in commonly used modes, which are independent of their databases.

Most recently, Chuang et al. [13] propose memory-constraint *top-k* mining (MTK) realizing the concept of the δ-stair search to retrieve *top-k* (closed) frequent patterns. MTK is devised for mining frequent itemsets and closed itemsets, without specifying the subtle *min_sup*. Although MTK tries to avoid *min_sup* to be specified in advanced however, it still needs to be computed internally. One may obtain *top-k* frequent itemsets by initially setting the *min_sup* equal to zero, and then raising the *min_sup* equal to $sup_k(w)$, i.e., the support of the kth most frequent itemset after the wth database scan. The MTK also performs multiple database scans which are not desirable.

Liu et al. [34] propose a new concise representation of frequent itemsets, which uses a positive border instead of a negative border together with frequent generators to represent a complete set of frequent itemsets. The size of the new representation is guaranteed to be no larger than the total number of frequent itemsets; thus it is a true concise representation. Their results show that a positive border is usually orders of magnitude smaller than its corresponding negative border.

Journal: 10115 Article No.: 0363 MS Code: KAIS-2030.4 TYPESET DISK LE CP Disp.:2010/11/17 Pages: 30 Layout: Small

Keogh E. et al. [32] show that employing parameter-free data mining approach based on compression can obtain good results. They propose parameter-free solution to numerous data mining tasks including clustering, classification, and anomaly detection. However, they do not suggest any solution for parameter-free frequent pattern mining. Majority of methods reported in the literature, use *min_sup* threshold to discover frequent itemsets. Setting up such a *min_sup* depends very much on an analyst's heuristics.

4 The proposed framework

Most of the frequent pattern mining algorithms discussed in the preceding section try to improve efficiency by: (i) reducing the number of passes over the database, (ii) reducing the size of the database to be scanned in every pass, (iii) utilizing differing techniques for pruning of candidates, (iv) employing various complex data structures, and (v) devising efficient search techniques. However, little attention is paid to the major performance hindrance i.e., *min_sup* input parameter. In general, one is required to repeatedly execute an algorithm, heuristically tuning the value of *min_sup* over a wide range, until the desired result is obtained, inevitably a highly time-consuming process.

We present, in this paper, a novel approach that seeks to efficiently retrieve top few significant maximal frequent patterns without specifying the subtle *min_sup* parameter.

4.1 Frequent pattern clusters

We begin our discussion by noting that the majority of candidate generation methods are based upon a bottom–up approach, i.e., for each itemset of length k found to be frequent within the database, all of its supersets of length $(k+1)$ itemsets are generated and tested, until no further candidate frequent itemsets are found. In contrast, our approach is based upon a top–down mechanism that directly retrieves the top few MFIs after only generating length 2-itemsets. It is observed that the support count computed for the 2-itemsets is maximum and thereafter decreases monotonically for the length $(3, 4, \ldots, n)$ itemsets.

Through use of an association metric, it is possible to convert the 2-itemset support counts into proximity distances, which enables clustering of itemsets of strong association such that an item in a cluster is closer to one or more other items in the cluster than to any item not in the cluster. However, the quality of clustering, i.e. the ability to generate clusters of high intra-class similarity but low inter-class similarity, depends critically upon the choice of association metric. After review of existing association metrics (Sect. 2.3), we introduce a new metric, the *association ratio*, defined below (Definition 5), which we find better able to satisfy our requirements.

Definition 5 (*association ratio* (***AR***)) The *association ratio* between two items is an odds ratio, computed as $AR_{(i,j)} = P(x_i, x_j)/(1 - P(x_i, x_j))$. The probability of co-occurrence frequency of x_i and x_j is the relative frequency computed as $P(x_i, x_j) = f(x_i, x_j)/|D|$, where x_i and x_j are statistically independent.

The computation of the *association ratio* for the length 2-itemsets of the sample dataset of Table 1 is shown in Table 4 below.

The first column of Table 4 lists all length 2-itemsets, the second column shows their frequency count probabilities, and third column shows the computed *association ratios*. The clusters of frequent itemsets can be now constructed using association ratio values. The XY-scatter plot is an effective way to visualize clusters of paired attributes. The view of an

Journal: 10115 Article No.: 0363 MS Code: KAIS-2030.4 TYPESET DISK LE CP Disp.:2010/11/17 Pages: 30 Layout: Small

Table 4 Computation of association ratio using sample database of Table 1

Itemsets	$P(X_i, X_j)$	$\frac{P(X_i,X_j)}{(1-P(X_i,X_j))}$
A B	0.7000	2.33
A C	0.2000	0.25
A D	0.4000	0.67
A E	0.1000	0.11
A F	0.8000	4.00
B C	0.4000	0.67
B D	0.5000	1.00
B E	0.3000	0.43
B F	0.8000	4.00
C D	0.3000	0.43
C E	0.3000	0.43
C F	0.3000	0.43
D E	0.2000	0.25
D F	0.5000	1.00
E F	0.2000	0.25

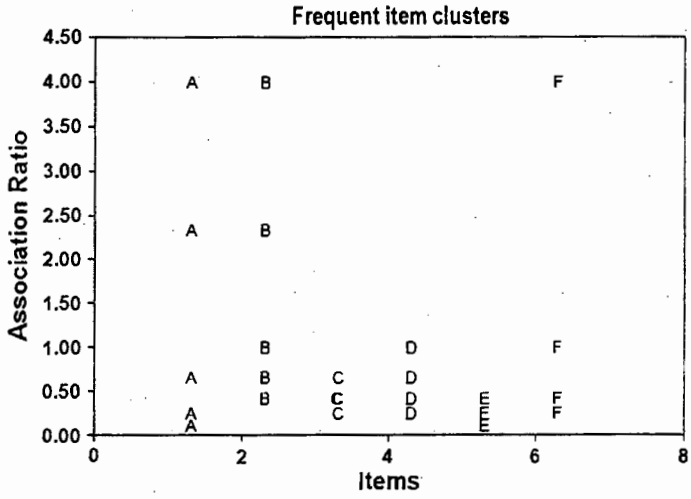


Fig. 2 The view of XY-scatter plot

XY-scatter plot for Table 4 is shown in Fig. 2, with the x-axis representing database items and the y-axis displaying the association ratios of the 2-itemsets. The most frequent itemset i.e., {A, B, F} can clearly be located in the high-association ratio region of the scatter plot area, demonstrating the effectiveness of the association ratio in throwing highly correlated items to the top while relegating clusters of infrequent itemsets to the bottom of the plot area. Consequently, this approach enables the systematic discovery of maximal frequent itemsets by traversing the search space in a top–down fashion, all without resort to the use of the *min_sup* threshold.

Journal: 10115 Article No.: 0363 MS Code: KAIS-2030.4 TYPESET DISK LE CP Disp.:2010/11/17 Pages: 30 Layout: Small

4.2 *Top-k* maximal frequent itemsets

The recent advances in data mining [11,13,48] introduce another simple parameter, *top-k*, which refers to the k most frequent itemsets in the database. Instead of specifying the *min_sup*, one can specify the desired count (k) of frequent itemsets to be mined, a parameter more easily applicable and understood. It is noted, however, that the *top-k* parameter is subject to different treatments in different works—for example, Wang et al. [48] employ *top-k* parameter to mine frequent closed itemsets, Cheung and Fu [11] retrieves *k most frequent l*-itemsets, and Chuang et al. [13] uses *top-k* parameter for mining frequent itemsets and closed itemsets—whereas in this work the *top-k* parameter is employed to discover top few significant MFIs.

The problem of mining the top few significant maximal frequent itemsets without *min_sup* threshold can be viewed as discovering the top-most clusters of itemsets while scanning the scatter plot from high-association region toward low association region. This mining process should be quick, efficient, and independent of *min_sup* parameter, employing instead some alternate parameter to indicate the extent of mining required. Consequently, for our purposes, the *top-k* parameter refers to the k number of top MFIs in the database, leading us to some necessary definitions as follows:

Definition 6 (*top-most maximal frequent itemset* (***TMMFI***)) An itemset X is referred to as *top-most* maximal frequent itemset if for all other frequent itemsets Y the $AR(X) > AR(Y)$ and the $|X|$ and $|Y|$ is greater than or equal to three.

Definition 7 (*top-k maximal frequent itemset* (***TKMFI***)) Given the desired value of k, itemsets $(X_1, X_2, \ldots X_k)$ are *top-k* maximal frequent itemsets in Dsuch that the $AR(X_1) \geq AR(X_2) \geq \cdots \geq AR(X_k)$ and the $|X|$ is greater than or equal to three.

The purpose of specifying the parameter k is to limit the number of significant MFIs produced. Large values of k are undesirable to work with because this will generate a huge number of MFIs, as already occurs with most of the existing methods. Thus, the value of k should be kept as small as possible, i.e. between 2 and 9. Our goal in this paper is to discover the *top-most* and *top-k* maximal frequent patterns directly and efficiently without depending upon use of the *min_sup* parameter. This can be achieved by computing association among the items using their co-occurrence frequency counts.

4.3 Construction of association ratio graph

Certainly frequent pattern mining is of greater value when applied to transaction databases that are large in size, and the entire database cannot completely fit into the main memory. For this reason, it is desirable, as a preliminary step, to convert the database into some more compact representation to facilitate the mining process. Motivated by this idea, we develop in this section an intermediary data structure, the *association ratio-graph* (AR-graph), defined as follows:

Definition 8 (*association ratio graph* (***AR-graph***)) Let D be a database of T transactions over a set of items I. A graph $G = (V, E)$ consists of a set of vertices $V = \{v_i = i | i \in I\}$ and a set of undirected edges $E = \{e_{i,j} |$ edge between vertices $v_i and v_j \in V$, v_i and v_j appear in the same transaction$\}$. Additionally, every edge has some weight $w(e_{i,j}) = AR_{(i,j)}$. The resulting graph is called an *association ratio graph*.

In the AR-graph, each graph vertex v_i corresponds to an item i in the original database, and each undirected edge $e_{i,j}$, connecting vertices v_i and v_j, represents co-occurrence of items i

Journal: 10115 Article No.: 0363 MS Code: KAIS-2030.4 TYPESET DISK LE CP Disp.:2010/11/17 Pages: 30 Layout: Small

Algorithm 1 Construction of association ratio graph

```
Input: Dataset D
Output: AR-Graph G = (V, E)

1. Set V(G) = Φ and E(G) = Φ
2. for each transaction T1 ... Tk ∈ D {
3.       for i = 1 to n {                          /* where n= |Ti| */
4.           if (vi ∉ V(G)) {
5.               V(G) = V(G)Uvi
6.           }
7.           for j = i + 1 to n {
8.               if vj ∉ V(G) {
9.                   V(G) = V(G)Uvj
10.              }
11.              if ei,j ∉ E(G) {
12.                  E(G) = E(G)Uei,j
13.              }
14.              w(ei,j) + +
15.          }
16.      }
17. }
18. for each ei,j   ∈ E(G) {
19.     w(ei,j) = P(ei,j)/(1 − P(ei,j))          /* where P(ei,j) = f(xi, xj)/|D| */
20. }
```

and j in database transactions. Each edge $e_{i,j}$ has some positive weight, denoted as $w(e_{i,j})$, is initially the number of transactions that contain items i and j. The weight $w(e_{i,j})$ signifies the strength of association between the two database items. Larger edge weight indicates that the items are strongly associated and smaller edge weights represent weak association. The absence of an edge between the two vertices implies lack of association. Given a vertex numbering and the edge weights between the vertices, an undirected graph can be hashed to an adjacency matrix A. Only a single database scan is enough to construct the adjacency matrix. The adjacency symmetric matrix A of the AR-graph is defined as:

$$A_{i,j} = \begin{cases} w\left(e_{i,j}\right), & \text{if } e_{i,j} \text{ exist} \\ 0, & \text{otherwise} \end{cases}$$

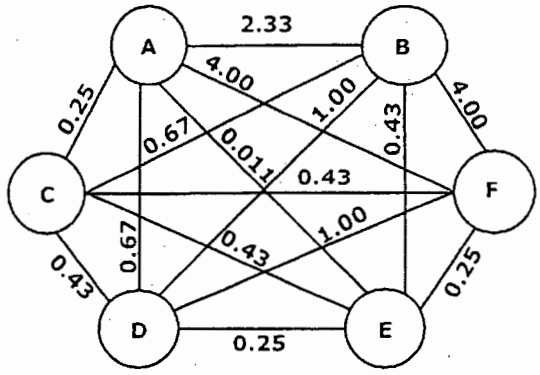
Based on the definition of AR-graph and association ratio measure, we now have the following AR-graph construction algorithm (Algorithm 1).

The algorithm scans the database one transaction at a time to build the AR-graph incrementally. For each database item, i, the algorithm instantiates a vertex, v_i, and for each co-occurrence of items i and j in a transaction, the algorithm instantiates an edge $e_{i,j}$, between vertices v_i and v_j, with initial edge value of 1 but incremented for each further co-occurrence of items i and j in later database transactions. After all edge values have been ascertained, these are converted to actual edge weights using the association ratio measure. An AR-graph constructed for Table 4 is shown in Fig. 3.

Given an AR-graph, *top-most* and *top-k* frequent itemsets can now be worked out separately. We discuss next mining the *top-most* maximal frequent itemset in Sect. 4.4 below, and the concept of mining *top-k* frequent itemsets following this in Sect. 4.5.

Journal: 10115 Article No.: 0363 MS Code: KAIS-2030.4 TYPESET DISK LE CP Disp.:2010/11/17 Pages: 30 Layout: Small

Fig. 3 An AR-graph for Table 4



4.4 Generation of top-most frequent itemset

The task of finding the *top-most* MFI can be defined as detecting a *maximal cycle* in the AR-graph. This is based on the observation that all the vertices representing items in the top-most MFI are linked together in the AR-graph through maximum-weight edges. Let D be a database of transactions and G the AR-graph corresponding to D. Then, every MFI in D corresponds to one or more maximal cycles in G. Consequently, the top-most MFI can be discovered by identifying vertices connected by edges with maximum weight. The AR-graph is traversed by selecting the edge with maximum weight at each step, until the cycle is discovered. All corresponding items to the vertices of the cycle represent the *top-most* maximal frequent itemset. The definition for the maximal cycle is as follows.

Definition 9 (*maximal cycle*) A *maximal cycle* is a path (containing at least three vertices) from a vertex to itself in such a way that at each vertex an edge having maximum weight is selected such that there is an edge between each vertex and its successor and all the vertices and edges are distinct.

Finding *top-most* maximal frequent itemset using maximal cycle is shown as follows (Algorithm 2).

Algorithm 2 constructs a maximal cycle in an AR-graph in order to discover the *top-most* MFI. Initially, all graph edges are sorted in decreasing order of their edge weights. At each step, the top-most edge $e_{i,j}$ not yet considered, having maximum weight, is selected from AR-graph along with its endpoint vertices (v_i and v_j) and added to the sub-graph (S). This process is repeated until a *maximal cycle* is formed in the sub-graph (Fig. 4). The maximal cycle is detected when both the endpoint vertices of a selected edge are found simultaneously in the sub-graph.

For the AR-graph of Fig. 3, the resultant maximal cycle is shown in Fig. 4. The given maximal cycle corresponds to the *top-most* maximal frequent itemset (MFI) i.e. {A, B, F}.

The sub-graph may or may not contain some fringe vertices to the maximal cycle, such fringe vertices being redundant in the uncovering of the maximal cycle. The fringe vertices can be easily pruned from the sub-graph to obtain the required maximal cycle, as all such vertices have degree one. This pruning step is performed using a priority queue (PQ). All sub-graph vertices are arranged in the PQ in increasing order of vertex degree. At each step, a vertex is drawn and removed from the PQ if its degree is one, followed by recalculation of the degrees of the remaining vertices and updating of the PQ. The *for* loop terminates after

Journal: 10115 Article No.: 0363 MS Code: KAIS-2030.4 TYPESET DISK LE CP Disp.:2010/11/17 Pages: 30 Layout: Small

Algorithm 2 Finding *top-most* maximal frequent itemset

Input: *AR-graph* $G = (V, E)$
Output: *Maximal Cycle PQ*

1. *Initialize the sub-graph* $S = (V, E)$ *by setting* $V(S) = \Phi$ *and* $E(S) = \Phi$
2. *Sort all edges* $E(G)$ *by weight in decreasing order*
3. $e_{i,j}$ = *edge with most weight not yet considered from the set* $E(G)$
4. $[v_i, v_j]$ = *end point vertices connected by* $e_{i,j}$
5. $V(S) = V(S) \cup v_i \cup v_j$
6. *Degree* $(v_i) + +$: *Degree* $(v_j) + +$
7. $E(S) = E(S) \cup e_{i,j}$
8. ***repeat***
9. $e_{i,j}$ = *edge with most weight not yet considered from the set* $E(G)$
10. $[v_i, v_j]$ = *end point vertices connected by* $e_{i,j}$
11. $V(S) = V(S) \cup v_i \cup v_j$
12. *Degree*$(v_i) + +$: *Degree*$(v_j) + +$
13. $E(S) = E(S) \cup e_{i,j}$
14. ***until*** $(v_i \in V(S) || v_j \in V(S))$
15. *Arrange all n vertices in a priority queue PQ in decreasing order of degree*
16. ***for*** $(i = 1; i <= n; i + +)$ {
17. ***if*** (*Degree*$(v_i) < 2$) {
18. *remove*(PQ, v_i)
19. *recalculate degree of remaining vertices*
20. *update*(PQ)
21. }
22. }
23. *return (PQ)* /* *PQ is left with only those vertices having degree greater than or equal to 2 representing MFI.* */

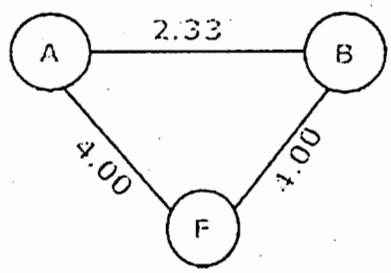


Fig. 4 Maximal cycle representing the *top-most* frequent itemset {A, B, F}

examining the degree of all vertices of the sub-graph, leaving the PQ populated with only the maximal cycle vertices corresponding to items of the *top-most* maximal frequent itemset (MFI).

4.5 Generation of *top-k* maximal frequent itemsets

We turn now to the task of discovering a desired number of significant MFIs. We begin, as above, by drawing the AR-graph and tracing maximal cycles subsequently. The procedure is supplemented with the introduction of the ASD-tree for finding multiple alternate paths for the underlying maximal cycle. The node values along these paths produced by ASD-tree can be picked up to discover the *top-k* maximal frequent itemsets (TKMFI). The pseudo-code is as follows (Algorithm 3).

Journal: 10115 Article No.: 0363 MS Code: KAIS-2030.4 TYPESET DISK LE CP Disp.:2010/11/17 Pages: 30 Layout: Small

Algorithm 3 **Finding *top-k* maximal frequent itemsets**

Input: *Database D, Desired number of MFIs k*
Output: *Top k MFIs*

```
    /* initialize global variables */
1. AR-Graph G = (V, E); V(G) = Φ; E(G) = Φ;
2. M[n, n] = Φ /* to maintain frequency count where n is the number of items in the database*/
3. List L[n, 3] = Φ /* to maintain 2-itemsets association ratio list */
4. T_k[k, 2] = Φ /* the repository to maintain top-k MFIs and their weights*/
   /* Scan database D to generate all 2-itemsets and computing frequency counts*/
5. for each transaction T_i ∈ D
6.         for i = 1 to n
7.             for j = i + 1 to n
8.                 M[i, j] = M[i, j] + 1
9.             end for
10.        end for
11. end for
   /* Update frequency counts to association ratios */
12. for i = 1 to n
13.         for j = i + 1 to n
14.             M[i, j] = M[i, j]/|D|
15.             M[i, j] = M[i, j]/(1 − M[i, j])
16.         end for
17. end for
18. Sort M in decreasing order of association ratio to produce2-itemset association ratio list L
19. for each row ∈ L
20.     [v1, v2] = L.read
21.     if (addition of edge e1,2 between v1 and v2 does not create a cycle in G)
22.         Add v1 and v2 as vertices to G
23.         Create an edge e1,2 between v1 & v2 and assign sequence number as edge weight
24.     else
25.         asd_tree (G, v1, v2) /* call procedure asd_tree by designating v1 as source vertex and v2
               as destination vertex*/
26.         if (T_k contains knumber of MFIs)
27.             stop
28.         end if
29.     end if
30. end for
```

Algorithm 3 follows the same basic approach outlined in Algorithms 1 and 2 above to produce the undirected AR-graph: (i) the database is scanned to obtain all length 2-itemsets from each transaction; (ii) frequency counts are hashed to a symmetric $|I|$-by-$|I|$ matrix M, row and column indices of M corresponding to item labels in the database; (iii) frequency counts are converted to proximity distances using the *association ratio* measure; (iv) all length 2-itemsets and their association ratios are arranged in the length 2-itemset association ratio list L comprising three columns (first item, second item, association ratio) and sorted in decreasing order of association ratio; and (v) the undirected AR-graph is constructed iteratively using each row from L to create an edge between vertices representing the items in columns 1 and 2. Noting that, we have used the sequence numbers of sorted rows in L as an alternate value for edge weights instead of association ratio.

Algorithm 3 discovers the *top-k* MFIs incrementally by examining new vertices and edges as they are added to the AR-graph. For each row of L from which an edge is to be introduced into the AR-graph, the following three cases exist:

Journal: 10115 Article No.: 0363 MS Code: KAIS-2030.4 TYPESET DISK LE CP Disp.:2010/11/17 Pages: 30 Layout: Small

Case 1: Both vertices v_1 and v_2 are not found in the AR-graph. In this case, v_1 and v_2 are added to the graph and an edge is created between them.

Case 2: The vertex v_1 is found and v_2 is not found or vice versa. In this case, the vertex not found is added to the graph and the edge is created.

Case 3: Both vertices v_1 and v_2 are found in the AR-graph. In this case, an ASD-tree (defined below as Definition 10) is created, following which the edge between v_1 and v_2 is added to the graph.

The first two cases are simple. In Case 3, one of the two vertices is designated as source vertex (v_s) and another is designated as destination vertex (v_t). The AR-graph is then required to be traversed to find all paths from source vertex v_s to destination vertex v_t in order of increasing length and weight. In graph theory, the computation of shortest paths is an important task and a long studied problem, where a path is searched between two given vertices in a graph such that the sum of the weights of its constituent edges is minimized. The shortest-path algorithms have diverse applications within related disciplines such as operations research, management science, geography, transportation, and computer science.

Several algorithms have been proposed to find shortest paths efficiently with a number of different variants such as: (i) the ***single-pair shortest-path problem*** is to find a shortest path from a source vertex v_s to a destination vertex v_t in the graph. The most well-known algorithm is due to Bellman [6]. (ii) The ***single-source shortest-path problem*** is to find shortest paths from source vertex v_s to all other vertices in the graph. The most common algorithms are those by Bellman [6], Dijkstra [15], and Hart et al. [27]. (iii) The ***single-destination shortest-path problem*** is to find shortest paths from all vertices in the graph to a single-destination vertex v_t. This can be reduced to the single-source shortest-path problem by reversing the edges in the graph. (iv) The ***all-pairs shortest-path problem*** is to find shortest paths between every pair of vertices v, v' in the graph. The most well-known algorithms are due to Floyd [20] and Johnson [30]. (v) The ***k shortest paths problem*** is to find the k paths connecting a given source–destination pair in the graph with minimum total length. The most efficient known algorithm to solve this problem is due to Eppstein [17]. The shortest-path algorithms compute not only shortest-path weights, but also produce a list of vertices on shortest paths as well. Most of the algorithms use breadth-first trees for the representation of shortest paths. However, none of them deal with finding *all paths* from a source vertex v_s to a destination vertex v_t directly. Given an undirected graph G with nonnegative weights and two vertices v_s and v_t, our problem needs to discover all paths from v_s to v_t in increasing order of length. We require that the paths be *simple* (loop free). We propose an all-path-source-destination tree (ASD-tree) that finds all paths connecting a given source–destination pair in the AR-graph using breadth-first search. The ASD-tree produces all paths starting at vertex v_s, visiting some intermediate vertices, and ending at vertex v_t. The ASD-tree is defined as follows.

Definition 10 (*all-path-source-destination tree* (***ASD-tree***)) Given an AR-graph, a source vertex (v_s) and a destination vertex (v_t), an ASD-Tree is a general tree T with a finite set of one or more nodes such that there is one designated node $r = v_s$, called root of T, and the remaining nodes are partitioned into $n \geq 0$ disjoint subsets $T_1, T_2, \ldots, T_n$, each of which is a tree, and whose roots are children of r such that all nodes in each path (from root node to leaf nodes) are distinct and a leaf node having v_t has no children.

Given v_s and v_t in an AR-graph G, let T be an all path source-to-destination tree (ASD-tree) with v_s as a root node. We next insert all vertices of AR-graph reached from v_s in T as child nodes of the root node. Each child node itself becomes the parent to further child nodes by examining all vertices connected to it in the AR-graph. A child node is only inserted in T if

Journal: 10115 Article No.: 0363 MS Code: KAIS-2030.4 TYPESET DISK LE CP Disp.:2010/11/17 Pages: 30 Layout: Small

Fig. 5 Array T[$n \times 4$] as an ASD-tree data structure

	Node	Parent	Weight	Path
0	D	null	0	
1	B	0	4	
2	A	1	3	
3	F	1	2	Y
4	F	2	1	Y

it does not already exist in the path from its parent to the root node of the tree. The leaf node holding destination vertex v_t need not be extended any further. The paths of the ASD-tree from root node v_s to leaf nodes containing the value v_t correspond to all source-to-destination paths in the AR-graph.

An ASD-tree (T) can be implemented as a dynamic four column array structure, as shown in Fig. 5. The first column of the array stores the value of each tree node, the second column contains a pointer back to the node's parent (thus preserving the child-parent link), the third column contains the edge weight, and a Boolean fourth column is marked true if the node is equal to v_t. The construction of T ends when no leaf node can be further extended, according to the properties given in Definition 10. We now need to traverse only those leaf nodes which are marked true to discover all source-to-destination paths. The nodes along a particular source-to-destination path in the ASD-tree correspond to all items of a particular MFI; the weight of each MFI can also be computed by adding the edge weights along the path in the tree.

Each discovered MFI with its associated weight is maintained in T_k, which is sorted in increasing order of weight indicating the decreasing level of significance, since sequence numbers are being used in place of the association ratios (i.e. sequence number 1 represents the highest association ratio). The algorithm terminates when $|T_k| = k$.

The procedure to construct the ASD-tree is as follows (Algorithm 4).

Running example of mining ***top-k*** *MFIs:* We perform Algorithm 3 on the sample database of Table 1 to retrieve the top-five ($k = 5$) MFIs systematically. After performing the database scan, all length 2-itemsets and their association ratios are arranged in decreasing order of association ratio as shown in Table 5. A step-by-step construction procedure of the AR-graph using the length 2-itemsets association ratio list and ASD-trees is shown in Fig. 6.

The *for each* loop fetches rows from the length 2-itemsets association ratio list, L, iteratively. The itemset {A, F} is picked in the first step. This follows Case 1 discussed above. The items A and F are made AR-graph vertices and an edge $e_{A,F}$ is created. The sequence number one is assigned as weight to the edge $e_{A,F}$. Fetching next row from L provides items B and F. This follows Case 2. A new vertex B is added to the graph and a weighted edge $e_{B,F}$ is created. The next row of L provides items A and B. This edge $e_{A,B}$ is able to create the first cycle in the AR-graph as shown in Fig. 6a. This follows Case 3 as discussed above. After designating the vertex A as source vertex (v_s) and vertex B as destination vertex (v_t), the ASD-tree procedure is then called to construct an ASD-tree to find all paths from the designated source-to-destination in the AR-graph. Alternate assignments of vertices as v_s or v_t, would cause the trees to be drawn somewhat differently but would nevertheless lead to the same results.

The ASD-tree is implemented using the ($n \times 4$) dynamic array of Fig. 5. The root node of the tree is assigned the value of the source vertex (v_s), in our example, A, and its parent column is marked as null. We next search AR-graph to find all vertices connected to vertex

Journal: 10115 Article No.: 0363 MS Code: KAIS-2030.4 TYPESET DISK LE CP Disp.:2010/11/17 Pages: 30 Layout: Small

Algorithm 4

```
Procedure: asd_tree(AR-graph G, v_x, v_t)
/* To construct all-path-source-to-destination tree (ASD-tree) */
1. T[n, 4] = Φ;
2. T[0, 1] = v_x; T[0, 2] = null; T[0, 3] = 0; T[0, 4] = false;
3. V[n] = Φ; W[n] = Φ;
4. T_k = Φ
5. i = 0
6. do while (T[i, 1] not null)
7.       x = T[i, 1]
8.       find x in G
9.       m = 1
10.      for each edge e_i ∈ x
11.          V[m] = V(e_i)
12.          W[m] = Weight (e_i)
13.          m++
14.      end for
15.      j = 1
16.      for each vertex v_i ∈ V
17.          if (T[i, 1] ≠ v_t && v_i is not present in the path from T[i, 1] to root)
18.              T[i + j, 1] = v_i
19.              T[i + j, 2] = i
20.              T[i + j, 3] = W[v_i]
21.              if (T[i + j, 1] = v_t)
22.                  T[i + j, 4] = True
23.              else
24.                  T[i + j, 4]=False
25.              end if
26.              j++
27.          end if
28.  end for
29.  i++
30. end do
      /* Traverse all nodes along each '-destination path in the ASD-tree to retrieve MFIs */
31. for i = 1 to n
32.      mfi = Φ
33.      w = 0
34.      if (T[i, 4])
35.          mfi = mfi ∪ T[i, 1]
36.          par = T[i, 2]
37.          w = w + T[i, 3]
38.          do while (par is not null)
39.              mfi = mfi ∪ T [par, 1]
40.              w = w + T[par, 3]
41.              par = T[par, 2]
42.          end do
43.      end if
44.      T_k = T_k ∪ (mfi, w)
45. end for
```

A giving the vertex set {F}. The vertex F is only inserted as a child node to node A, if F does not exist at any node in the path from present node to root node. Since F is not found thus it is added as child of A. Now the same processing is carried out for the next node in the tree i.e. F. Once again AR-graph is searched to find all vertices connected to the vertex F, giving the vertex set {A, B}. The item A must be discarded as it already exists in the path from node F, to root node, A. As item B is not found in the path from node F to root node

Journal: 10115 Article No.: 0363 MS Code: KAIS-2030.4 TYPESET DISK LE CP Disp.:2010/11/17 Pages: 30 Layout: Small

Table 5 2-itemsets association ratio list

S.No.	Itemsets	AR
1	{A, F}	4.00
2	{B, F}	4.00
3	{A, B}	2.33
4	{B, D}	1.00
5	{D, F}	1.00
6	{A, D}	0.67
7	{B, C}	0.67
8	{B, E}	0.43
9	{C, D}	0.43
10	{C, E}	0.43
11	{C, F}	0.43
12	{A, C}	0.25
13	{D, E}	0.25
14	{E, F}	0.25
15	{A, E}	0.01

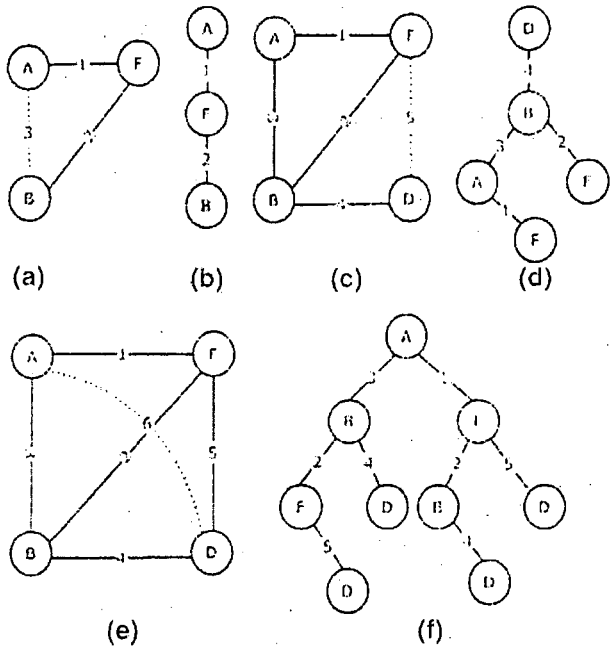


Fig. 6 The illustrative example of discovering top five MFIs using AR-graph

this is added as a child node of F. Here, the path column of the ASD-tree array is set to true as B is the destination, and this path also reaches termination according to the ASD-tree definition discussed above, thus completing the tree construction process as shown in Fig. 6b. The ASD-tree thus produced exposes a single cycle only, the first MFI {A, F, B} of weight 3, found by traversing the single path from destination node to root node (root and destination nodes gray-shaded in these figures).

Journal: 10115 Article No.: 0363 MS Code: KAIS-2030.4 TYPESET DISK LE CP Disp.:2010/11/17 Pages: 30 Layout: Small

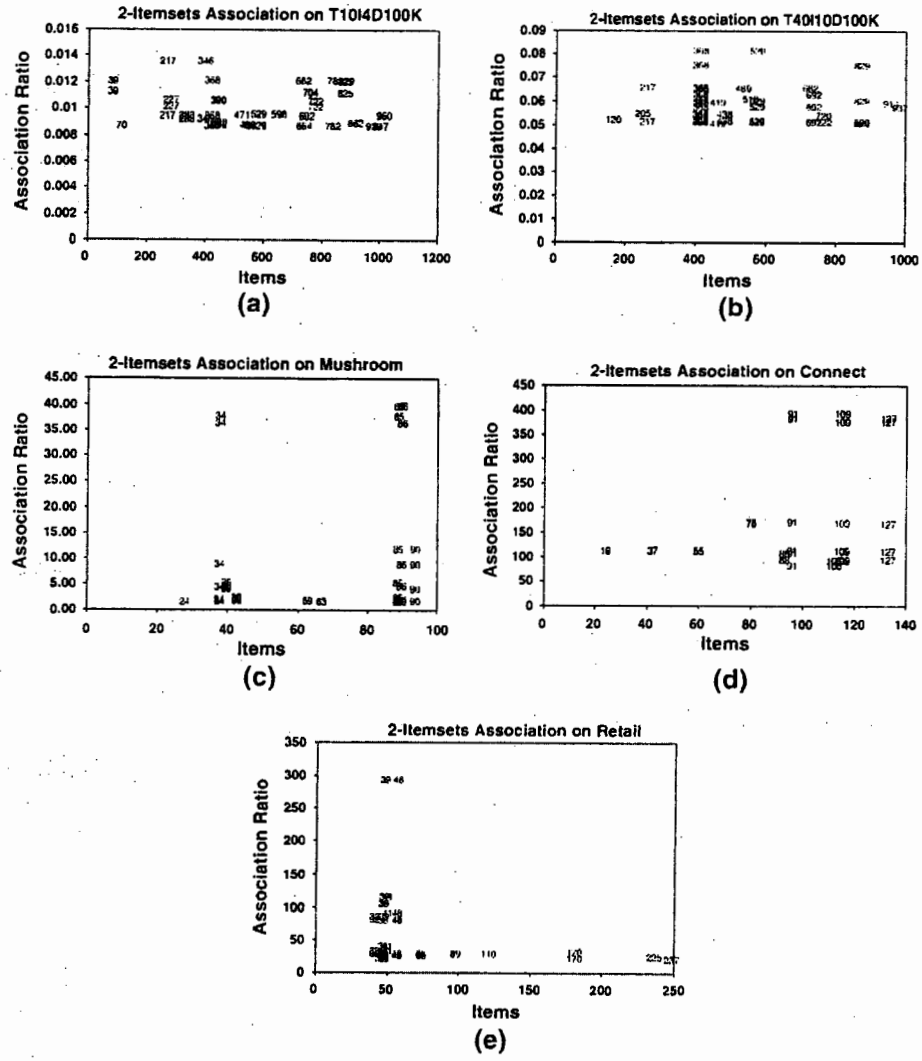


Fig. 7 The frequent pattern clusters drawn on standard datasets

Edge $e_{A,B}$ is now inserted into the graph and the algorithm continues to fetch the next row from L. The addition of the fourth edge ($e_{B,D}$) does not cause any new cycle in the AR-graph (Fig. 6c), but the insertion of the fifth and sixth edges ($e_{D,F}$ and $e_{A,D}$) does create new cycles (Fig. 6c, e) requiring construction of subsequent ASD-trees (Fig. 6d, f). The first of these two ASD-trees (Fig. 6d) exposes two new cycles in the graph corresponding to MFIs {D, B, F} and {D, B, A, F} of weights 6 and 8 respectively. The second of these two ASD-trees (Fig. 6f) exposes four new cycles corresponding to MFIs {A, B, F, D}, {A, B, D}, {A, F, B, D} and {A, F, D} of weights 10, 7, 7 and 6, respectively.

All discovered MFIs are maintained in the top-k result table (T_k) shown in Fig. 7. Duplicate itemsets are eliminated, retaining the version with the smallest weight. The complete set of top-5 MFIs is shown in Table 6 arranged in decreasing order of significance.

Journal: 10115 Article No.: 0363 MS Code: KAIS-2030.4 TYPESET DISK LE CP Disp.:2010/11/17 Pages: 30 Layout: Small

Table 6 *Top-k* MFI results sorted by the weight

S.No.	MFI	Weight
1	{A, F, B}	3
2	{D, B, F}	6
3	{A, F, D}	6
4	{A, B, D}	7
5	{A, F, B, D}	7

Table 7 Dataset parameters

Dataset	Number of items	Number of transactions	Average transaction size
T10I4D100K	1,000	1,00,000	10
T40I10D100K	1,000	1,00,000	40
Mushroom	120	8,124	23
Connect	130	67,557	43
Retail	16,470	88,162	10

5 Empirical evaluation

After illustrating the effectiveness of our approach using a simple example in the previous section, we present here experimental results obtained from running the algorithm on standard datasets against other popular frequent pattern mining algorithms. Our experiments were conducted using a 3.2-GHz IBM-compatible PC with 1 GB of memory, running Windows XP.

In selecting alternate algorithms for performance comparison, we note that a large number of algorithms are available for mining FI, CFI, and MFI. Apriori [3] is the first and the most influential algorithm for frequent pattern mining and is the model for many other proposed algorithms. We opted for the Borgelt's [7] implementation of Apriori that employs a prefix tree to organize the counts for the itemsets. Other state-of-the-art algorithms for mining MFIs are DepthProject [1], Mafia [10], and GenMax [21], though these share much in common, as noted by Tianming et al. [43]: search the item subset lattice in a depth-first way, apply look-a-head pruning and dynamic reordering to reduce the search space and use compression techniques for fast support counting. For our experiments, we considered it sufficient to select Mafia from these for performance analysis purposes. As regards datasets, we chose both real and synthetic datasets that have been widely used in frequent itemsets mining studies. Selected characteristics of the chosen datasets are shown in Table 7.

5.1 Frequent pattern clusters on standard datasets

The effectiveness of our top–down approach to discover the top few maximal frequent itemsets can also be demonstrated for the standard datasets of Table 7. We have provided below, in Fig. 7, XY-scatter plots illustrating clustering of the most significant MFIs in the high-association ratio regions of each plot. All plots are drawn for only the top 20 pair of itemsets to avoid the clutter. The top-most clusters {217, 346}, {368, 529, 829}, {34, 85, 86}, {91, 109, 127} and {39, 48} are clearly visible at the top of the plots in Fig. 7a, b, c, d and e, respectively. In each of the plots, clusters of significant MFIs are few in the high-association ratio region but excessive overlapping of itemsets further down the plots represents the typically large

Journal: 10115 Article No.: 0363 MS Code: KAIS-2030.4 TYPESET DISK LE CP Disp.:2010/11/17 Pages: 30 Layout: Small

numbers of infrequent items with low association ratio found in such databases. Note that both T10I4D100K and T40I10D100K datasets show numerous two itemset clusters. This is because both datasets have very high concentration of length 2-itemsets.

5.2 Experiments on mining top-most MFI

We now investigate the efficiency of Algorithm 2, presented in Sect. 4.4, used to discover the top-most MFI (TMMFI) on the chosen standard datasets. We confine ourselves to comparing CPU execution times only, though it should be noted that existing methods, such as Apriori and Mafia, require considerable additional manual time also, after each run, to review results and select a new *min_sup* value for the next run, until the desired results are obtained.

Table 8 compares execution times for Apriori, Mafia and our Algorithm 2 (TMMFI). For each dataset, Apriori and Mafia were executed a number of times with *min_sup* values ranging from 1 to 99.95% in order to discover the top-most MFI. Results produced by Apriori and Mafia naturally contain a huge number of itemsets at low *min_sup* values: for example, Apriori and Mafia generated respectively 65,236 and 21,692 itemsets from the T40I10D100K dataset when run at 1% *min_sup*. Discovering the one top-most MFI using Apriori or Mafia requires trawling through thousands of results.

Our approach performs very well in discovering the one *top-most* MFI, which is required to be executed once only as there is no need for iterative tuning of the *min_sup* parameter. Fig. 8 shows the sub-graphs produced by Algorithm 2 (TMMFI) for the standard datasets. In each sub-graph, the top-most MFI can be identified by traversing the exposed maximal cycles. Sub-graphs generated for the Mushroom (Fig. 8c) and Connect (Fig. 8d) datasets require no further pruning, whereas T10 (Fig. 8a), T40 (Fig. 8b) and Retail (Fig. 8e) make use of the pruning step to expose the maximal cycles. The top-most MFI results and execution time of TMMFI for these datasets are presented in the last two columns of Table 6 above.

In comparing the performance of TMMFI with Apriori and Mafia, it is important to note that, even if we succeed in accurately setting the *min_sup* parameter, Apriori and mafia provide no clear way to easily discover the top-most MFI from the itemset results. Although any particular individual run of Apriori or Mafia may execute faster than TMMFI, however our approach is, on balance, much more efficient when compared to the accumulated time for the multiple Apriori and mafia executions required in order to reach the desired results, even if the time taken for manual tuning of the *min_sup* value is excluded from this latter measurement.

The performance comparison curves for Apriori, Mafia, and TMMFI on the five datasets are shown in Fig. 9. It is clear to see that our approach equals or considerably outperforms the other two algorithms on four out of five datasets. This performance gain is mainly due to the sparse character of the datasets and because the generation of length 3-itemsets, 4-itemsets ... *n*-itemsets is avoided altogether. With Retail, the performance of TMMFI is worse than the other two algorithms primarily because this dataset is not as sparse as the others. The TMMFI has to construct a very large 2-itemset matrix due to large number of items in the database. The results produced by all the three algorithms are in agreement with each other except in case of Retail where Mafia produces a different MFI.

5.3 Experiments on mining *top-k* MFI

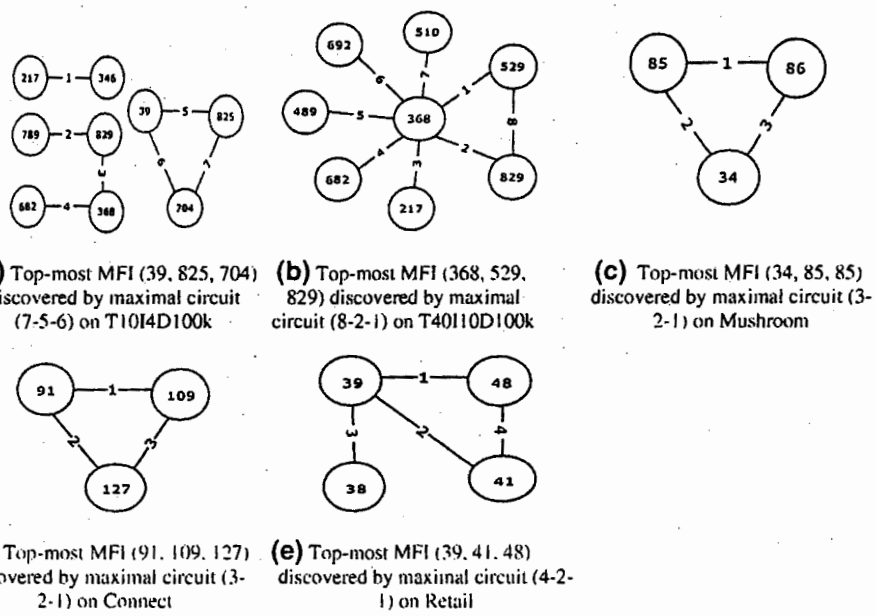
We now present the results of execution of Algorithm 3 (TKMFI) on standard datasets (Table 7) to retrieve the *top-k* MFIs. The TKMFI is tested for k set to 5. A view of the final AR-graphs generated by TKMFI algorithm for each dataset is shown in Fig. 10.

Journal: 10115 Article No.: 0363 MS Code: KAIS-2030.4 TYPESET DISK LE CP Disp.:2010/11/17 Pages: 30 Layout: Small

Table 8 Results produced by the three algorithms on standard datasets

Datasets	Apriori				Mafia				TMMFI (Algo-2)	
	Min. supp. (%)	# MFI	Top-Most MFI	Time (S)	Min. supp. (%)	# MFI	Top-Most MFI	Time (S)	Top-Most MFI	Time (S)
T10I4D100K	1	385	39, 704, 825	2.61	1	370	39, 704, 825	18	39, 704, 825	2.5
	2	155	368	2.24	2	155	368	12		
	10	0			10	0				
	20	0			20	0				
T40I10D100K	1	65236	368, 529, 829	26.87	1	21692	368. 529, 829	98	368. 529, 829	16.65
	2	2293	368, 529. 829	11.84	2	2015	682 489 368	59		
	10	82	368	8.01	10	82	368	43		
	20	5	368	6.98	20	5	368			
Mushroom	20	1197	34. 85. 86	1.67	20	158	24. 34, 3 9 85. 86, 90	2	34, 85, 86	1.63
	45	83	34, 85, 86	0.4	45	18	24. 34. 85. 86. 90	1		
	70	12	34. 85. 86	0.33	70	1	34. 36, 85. 86. 90	2		
	95	4	34. 85. 86	0.36	95	1	34. 85. 86	2		
Connect	98	21	55. 75, 127. 109, 91	4.79	98	21	55 75 127 109 91	23	91, 109, 127	4.17
	99	11	75. 91, 109. 127	4.36	99	11	75 127 109 91	23		
	99.94	4	91. 109, 127	4.71	99.40	4	127 109 91	23		
	99.95	2	91. 109. 127	4.42	99.50	2	127 109 91	22		
Retail	1	159	39, 41, 48	2.53	1	78	39, 48, 89	11	39. 41. 48	89
	2	55	39. 41, 48	2.36	2	20	32, 39. 48	10		
	10	9	39	2.36	10	5	39. 48	11		
	20	3	39	2.29	20	1	39, 48	11		

Springer
Journal: 10115 Article No.: 0363 MS Code: KAIS-20304 TYPESET DISK LE CP Disp.:2010/1/17 Pages: 30 Layout: Small



(a) Top-most MFI (39, 825, 704) discovered by maximal circuit (7-5-6) on T10I4D100k

(b) Top-most MFI (368, 529, 829) discovered by maximal circuit (8-2-1) on T40I10D100k

(c) Top-most MFI (34, 85, 85) discovered by maximal circuit (3-2-1) on Mushroom

(d) Top-most MFI (91, 109, 127) discovered by maximal circuit (3-2-1) on Connect

(e) Top-most MFI (39, 41, 48) discovered by maximal circuit (4-2-1) on Retail

Fig. 8 Discovering MFIs by maximal cycles on different datasets

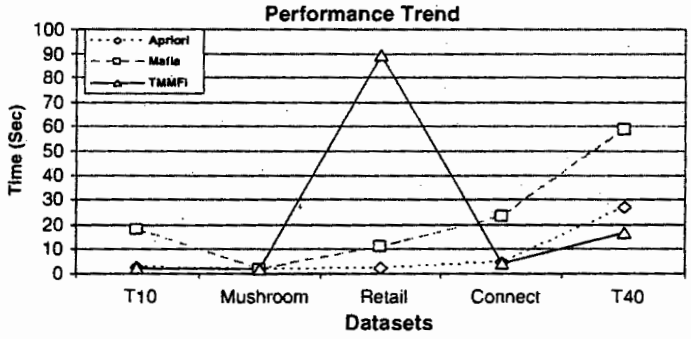


Fig. 9 Performance evaluation on different datasets

As discussed in Sect. 4.5, sequence number (representing the order in which each edge is added to the AR-graph) is used as each edge's weight. ASD-trees are constructed prior to insertion of each edge, if cycle formation is detected. The ASD-trees are used to expose the MFIs, whose items are represented by the nodes in each source-to-destination path. MFIs can then be ranked in order of weight by summing edge weights along each source-to-destination path in the ASD-tree.

As can be seen in Fig. 10, the top five MFIs are discovered by traversing the cycles formed in each AR-graph. Table 9 lists the first five cycles for each dataset and the MFIs they represent along with their weights. The results are ranked in increasing order of MFI weighting (corresponding to decreasing association ratio). In case of duplicate MFIs in the final list, only the lowest weighted MFI is retained. The majority of the algorithm's running time occurs during database scanning in order to generate the length 2-itemsets and to con-

Journal: 10115 Article No.: 0363 MS Code: KAIS-2030.4 TYPESET DISK LE CP Disp.:2010/11/17 Pages: 30 Layout: Small

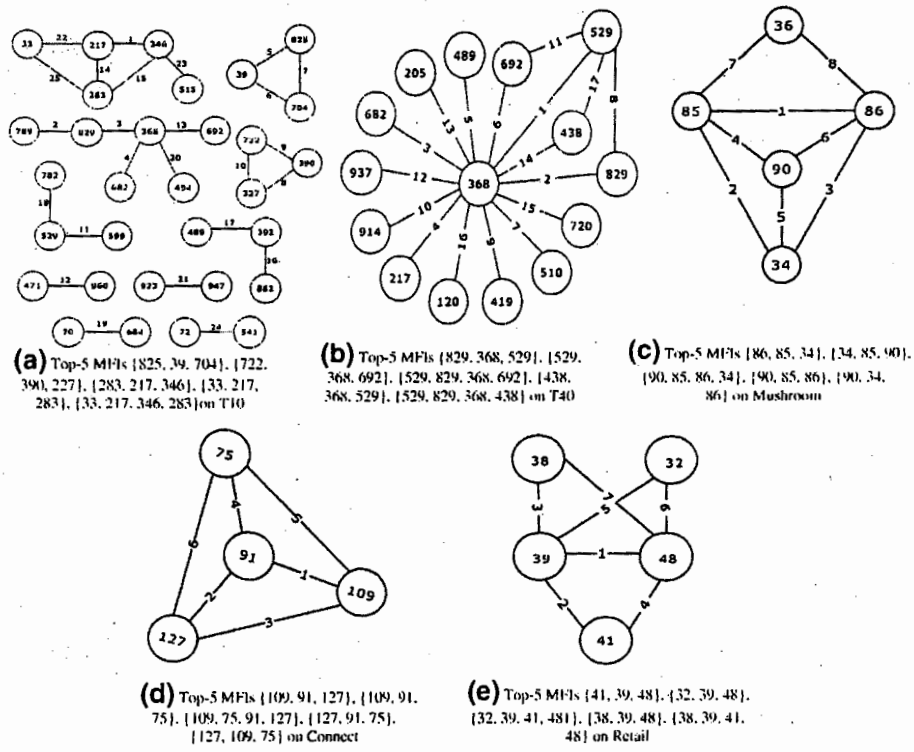


Fig. 10 AR-graphs showing top-5 MFI cycles on standard datasets

struct the support count matrix. Once the length 2-itemsets association ratio list is prepared, the algorithm rapidly constructs the AR-graph and ASD-tree reading only the top few rows from the list.

The technique that we have proposed in this paper demonstrates competitive efficiency in directly narrowing the search to only the top-most and *top-k* relevant maximal frequent itemsets by altogether avoiding superset generation and adjusting *min_sup* value. Mining without use of the *min_sup* threshold is the most prominent advantage of these algorithms making them a highly practical approach for mining *top-k* MFIs.

6 Conclusions

This paper presents a study of practically important data mining problem, namely mining maximal frequent patterns without *min_sup* threshold. To achieve this, we proposed two novel algorithms for mining *top-most* and *top-k* frequent itemsets. Based on effective graph-theoretic approach, we have been able to eliminate the requirement of specifying *min_sup* threshold by devising a novel search approach using ASD-tree to retrieve MFIs. Experiments performed on standard datasets verify the effectiveness of the proposed framework that can both achieve high efficiency and user friendliness, demonstrating their practical importance in the mining of maximal frequent itemsets. There is of course ample opportunity for future work. After embedding the approach in association rule mining, we plan to carry out the the-

Journal: 10115 Article No.: 0363 MS Code: KAIS-2030.4 TYPESET DISK LE CP Disp.:2010/11/17 Pages: 30 Layout: Small

Table 9 Retrieval of top-five MFIs on different datasets

Datasets	Cycle #	Edges making cycle	MFIs	MFI weight
T10I4D100K	1	(5, 6, 7)	{39, 704, 825}	18
	2	(8, 9, 10)	{227, 390, 722}	27
	3	(1, 14, 15)	{217, 283, 346}	30
	4	(14, 22, 25)	{33, 217, 346, 283}	61
	5	(22, 1, 15, 25)	{33, 217, 346, 283}	63
T4J0I10D100K	1	(8, 1, 2)	{368 , 529, 829}	11
	2	(11, 1 ,6)	{368, 529, 692}	18
	3	(11, 6, 2, 8)	{368, 692, 529, 829}	27
	4	(17, 1, 14)	{368, 438, 529}	32
	5	(17, 8, 2, 14)	{368, 438, 529, 829}	41
Mushroom	1	(3, 2, 1)	{34, 85, 86}	6
	2	(5, 4, 2)	{34, 85, 90}	11
	3	(6, 1, 4)	{85, 86,90}	11
	4	(5, 4, 1, 3)	{34, 85, 86, 90}	13
	5	(6, 3, 5)	{34, 86, 90}	14
Connect	1	(3, 2, 1)	{91, 109, 127}	6
	2	(5, 4, 1)	{75, 91, 109}	10
	3	(6, 4, 2)	{75, 91, 127}	12
	4	(6, 5, 3)	{75, 109, 127}	14
	5	(5, 4, 2, 3)	{75, 91, 109, 127}	14
	1	(4, 2, 1)	{39, 41, 48}	7
	2	(7, 3, 1)	{38, 39, 48}	11
Retail	3	(6, 5, 1)	{32, 39, 48}	12
	4	(7, 3, 2, 4)	{38, 39, 32, 48}	15
	5	(6, 4, 2, 5)	{32, 39, 41, 48}	17

oretical analysis of this technique and examine its application to other interesting problems such as semantic image retrieval and annotation. This work can also be extended to other data mining tasks such as clustering and classification.

References

1. Agrawal R, Aggarwal CC, Prasad VVV (2000) Depth first generation of long patterns. In: ACM conference on knowledge discovery and data mining (SIGKDD), pp 108–118
2. Agrawal R, Imielinski T, Swami AN (1993) Mining Association Rules Between Sets of Items in Large Databases. In: Proceedings of the ACM SIGMOD international conference on management of data, pp 207–216
3. Agrawal R, Srikant R (1994) Fast algorithms for mining association rules. In: Proceedings of the 20th international conference on very large databases (VLDB'94). Santiago, Chile. Morgan Kaufmann, pp 487–499
4. Agrawal R, Srikant R (1995) Mining sequential patterns. In: Proceedings of the 11th international conference on data engineering, Taipei, Taiwan, pp 3–14
5. Bayardo RJ (1998) Efficiently mining long patterns from databases. In: Proceedings of ACM SIGMOD international conference on management of data
6. Bellman RE (1958) On a routing problem. Q Appl Math 16:87–90

Journal: 10115 Article No.: 0363 MS Code: KAIS-2030.4 TYPESET DISK LE CP Disp.:2010/11/17 Pages: 30 Layout: Small

7. Borgelt C (2004) Efficient implementations of apriori and eclat. In: 2nd workshop of frequent item set mining implementations, FIMI 2004. Brighton, UK
8. Brin S, Motwani R, Ullman J, Tsur S (1997) Dynamic itemset counting and implication rules for market basket data. In: Proceedings of ACM SIGMOD international conference on management of data. pp 255–264
9. Brin S, Motwani R, Silverstein C (1997) Beyond market baskets: generalizing association rules to correlations. In: Proceedings of ACM SIGMOD international conference on management of data. Arizona, pp 255–264
10. Burdick D, Calimlim M, Gehrke J (2001) Mafia: a maximal frequent itemset algorithm for transactional databases. In: 17th international conference on data engineering (ICDE). Heidelberg, Germany
11. Calders T, Dexters N, Goethals B (2007) Mining frequent itemsets in a stream. In: 2007 seventh IEEE international conference on data mining. pp 83–92
12. Cheung YL, Fu AW (2004) Mining association rules without support threshold: with and without item constraints. In: TKDE
13. Chuang K, Huang JL, Chen MS (2008) Mining top-k frequent patterns in the presence of the memory constraint. The VLDB J 17:1321–1344
14. Chung SM, Luo C (2008) Efficient mining of maximal frequent itemsets from databases on a cluster of workstations. Know Inf Syst 16:359–391
15. Dijkstra EW (1959) A note on two problems in connection with graphs. Numeriche Mathematik 1:269–271
16. El-Sayed M, Ruiz C, Rundensteiner EA (2004) FS-Miner: efficient and incremental mining of frequent sequence patterns in web logs. In: Proceedings of the 6th annual ACM international workshop on Web information and data management. WIDM '04
17. Eppstein D (1999) Finding the k shortest paths. SIAM J Comput 28(2):652–673
18. Exarchos TP, Tsipouras MG, Papaloukas C, Fotiadis DI (2009) An optimized sequential pattern matching methodology for sequence classification. Know Inf Syst 19:249–264
19. Fang G, Deng Z, Ma H (2009) Network Traffic Monitoring Based on Mining Frequent Patterns. In: Sixth international conference on fuzzy systems and knowledge discovery. pp 571–575
20. Floyd RW (1962) Algorithm 97: shortest path. Commun ACM 5(6):345
21. Gouda K, Zaki MJ (2005) GenMax: an efficient algorithm for mining maximal frequent itemsets. data mining and knowledge discovery, 11. Springer Science and Business Media, Netherlands, pp 1–20
22. Grahne G, Zhu J (2003) High performance mining of maximal frequent itemsets. In: Proceeding of the 6th SIAM international workshop on high performance data mining, pp 135–143
23. Gunopulos D, Khardon R, Mannila H et al (2003) Discovering all most specific sentences. ACM Trans Database Syst(TODS) 28(2)
24. H'ajek P, Havel I, Chytil M (1966) The GUHA method of automatic hypotheses determination. Computing 1:293–308
25. Han J, Cheng H, Xin D, Yan X (2007) Frequent pattern mining: current status and future directions. Data Min Knowl Disc 15:55–86
26. Han J, Pei J (2000) Mining frequent patterns by pattern growth: methodology and implications. SIGKDD Explor 2(2):14–20
27. Hart PE, Nilsson NJ, Raphael B (1968) A formal basis for the heuristic determination of minimum cost paths. IEEE Trans Syst Sci Cybern SSC4 4(2):100–107
28. Holt JD, Chung SM (2001) Multipass algorithms for mining association rules in text databases. Know Inf Syst 3(2):168–183
29. Holt JD, Chung SM (2002) Mining association rules using inverted hashing and pruning. Inf Process Lett 83(4):211–220
30. Johnson DB (1977) Efficient algorithms for shortest paths in sparse networks. J ACM 24(1):1–13
31. Khan A, Khan K, Baharudin BB (2009) Frequent patterns mining of stock data using hybrid clustering association Algorithm. In: International conference on information management and engineering. pp 667–671
32. Keogh E, Lonardi S, Ratanamahatana CA (2004) Towards parameter-free data mining. In: KDD'04
33. Liu B, Hsu W, Ma Y (1998) Integrating classification and association rule mining. In: Proceedings of the 4th International Conference on knowledge discovery and data mining. New York, pp 80–86
34. Liu G, Li J, Wong L (2008) A new concise representation of frequent itemsets using generators and a positive border. Know Inf Syst 17:35–56
35. Mannila H, Toivonen H, Verkamo AI (1997) Discovery of frequent episodes in event sequences. Data Min Knowl Discov 1(3):259–289
36. Pang-Ning T, Vipin K, Jaideep S. (2004) Selecting the right objective measure for association analysis. Inf Syst 29:293–313

Journal: 10115 Article No.: 0363 MS Code: KAIS-2030.4 TYPESET DISK LE CP Disp.:2010/11/17 Pages: 30 Layout: Small

37. Park JS, Chen MS, Yu PS (1997) Using a hash-based method with transaction trimming for mining association rules. IEEE Trans Knowl Data Eng 9(5):813–825
38. Pasquier N, Bastide Y, Taouil R, Lakhal L (1999) Discovering frequent closed itemsets for association rules. In: Proceeding of the 7th international conference on database theory (ICDT'99), Israel. pp 398–416
39. Pei J, Han J, Mao R (2000) CLOSET: an efficient algorithm for mining frequent closed itemsets. In: Proceedings of the 5th ACM SIGMOD workshop on research issues in data mining and knowledge discovery. pp 21–30
40. Piatetsky-Shapiro G (1991) Discovery, analysis and presentation of strong rules. In: Piatetsky-Shapiro G, Frawley W (eds) Knowledge discovery in databases. MIT Press, Cambridge, MA pp 229–248
41. Savasere A, Omiecinski E, Navathe S (1995) An efficient algorithm for mining association rules in large databases. In: Proceedings of the 21st VLDB conference. pp 432–444
42. Tan PN, Kumar V (2000) Interestingness measures for association patterns: a perspective. In: KDD 2000 workshop on postprocessing in machine learning and data mining. Boston, MA
43. Tianming H, Sam YS, Hui X, Qian F (2007) Discovery of maximum length frequent itemsets
44. Van Rijsbergen CJ (1979) Information Retrieval 2nd edn. Butterworths, London
45. Wan L, Liao J, Zhu X (2009) A frequent pattern based framework for event detection in sensor network stream data. In: Proceedings of the Third International Workshop on Knowledge Discovery from Sensor Data. SensorKDD '09
46. Wang H, Yang J, Wang W, Yu P (2002) Clustering by pattern similarity in large data sets. In: Proceedings of the ACM SIGMOD Intl. Conf. on Management of Data, Wisconsin. pp 394–405
47. Wang J, Han J, Pei J (2003) CLOSET+: searching for the best strategies for mining frequent closed itemsets. In: Proceeding of the ACM SIGKDD Intl. Conf. on knowledge discovery and data mining (KDD'03), Washington, DC. pp 236–245
48. Wang J, Han J, Lu Y, Tzvetkov P (2005) TFP: an efficient algorithm for mining top-k frequent closed itemsets. In: TKDE
49. Wu X, Zhang C, Zhang S (2004) Efficient mining of both positive and negative association rules. ACM Trans Information System 22(3):381–405
50. Yang G (2004) The complexity of mining maximal frequent itemsets and maximal frequent patterns. In: Proceeding of the ACM SIGKDD Intl. Conf. on knowledge discovery in databases (KDD'04), Seattle, WA, pp 344–353
51. Zaki MJ, Hsiao CJ (2002) CHARM: An efficient algorithm for closed itemset mining. In: Proceedings of the 2nd SIAM international conference on data mining. pp 12–28
52. Zhang S, Wu X, Zhang C, Lu J (2008) Computing the minimum-support for mining frequent patterns. Know Inf Syst 15:233–257

Author Biographies

Abdus Salam received his MSc in Computer Science from the Quaid-e-Azam University of Islamabad in 1987. He is currently a PhD student working under the supervision of Dr. Sikandar H. Khiyal at the International Islamic University, Islamabad. His research interests are in the areas of databases, data mining and information systems.

Journal: 10115 Article No.: 0363 MS Code: KAIS-2030.4 TYPESET DISK LE CP Disp.:2010/11/17 Pages: 30 Layout: Small

Dr. M. Sikandar Hayat Khayal is currently working as chairman department of computer sciences and software engineering at Fatima Jinnah Women University Pakistan. He served Pakistan atomic energy commission for 24 years. His research interests are numerical analysis of algorithm, theory of automata and theory of computation. He has more than 105 research publications published in National and International Journals and Conference proceedings. He is member of SIAM, ACM, Informing Science Institute, IACSIT.

Journal: 10115 Article No.: 0363 MS Code: KAIS-2030.4 ☐ TYPESET ☐ DISK ☐ LE ☐ CP Disp.:2010/11/17 Pages: 30 Layout: Small