2005

Structural advances for pattern discovery in multi-relational databases

Juveria Kanodia

Library Rights Statement

In presenting the thesis Structural Advances for Pattern Discovery in Multi-Relational Databases in partial fulfillment of the requirements for an advanced degree at the Rochester Institute of Technology, I agree that the Library shall make it freely available for inspection. I further agree that permission for copying as provided for by the Copyright Law of the U.S. (Title 17, U.S. Code) of this thesis for scholarly purposes may be granted by the Librarian. It is understood that any copying or publication of this thesis for financial gain shall not be allowed without my written permission.

I hereby grant permission to the RIT Library to copy my thesis for scholarly purposes.

Name: Juveria Kanodia

Date: 6/13/05

STRUCTURAL ADVANCES FOR PATTERN DISCOVERY IN MULTI-RELATIONAL DATABASES

BY

JUVERIA KANODIA

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
COMPUTER SCIENCE

ROCHESTER INSTITUTE OF TECHNOLOGY

MAY 2005

MASTER OF SCIENCE THESIS

OF

JUVERIA KANODIA

APPROVED:

Thesis Committee

Dr. Ankur Teredesai

Dr. Roger Gaborski

Dr. Vladimir Misic

ROCHESTER INSTITUTE OF TECHNOLOGY

MAY 2005

Acknowledgements

Firstly, I would like to thank my advisor Dr. A. Teredesai for constantly encouraging me to put in my best effort academically and in this thesis. He never ran out of time whenever I approached him with questions. I thank him for helping me with suggestions and ideas whenever I needed them and making data mining my favorite subject.

I would also like to thank Dr. R. Gaborski and Dr. V. Misic for being a part of my committee. Dr. Gaborski was a great source of knowledge and inspiration to me in his BIIS class. I thank Dr. Misic for having faith in me and my capabilities and appointing me as his grader for the database systems course. I have learnt a great deal from both the professors.

I would like to thank my entire committee to take out time from their busy schedules to read my thesis and attend my defense.

I would like to add a word of thanks to Shailesh for helping and motivating me during my thesis and for all his support. And I also thank all my friends and relatives for constantly being there for me.

And finally a great deal of thanks to my parents and my brother, Kabir, without whom, I wouldn't have been able to come to RIT to pursue Master's studies. Thank you for giving me the wonderful opportunity and encouraging me in all my endeavors and sharing with me all my successes and failures.

Thank you to my family, friends and all my teachers for making me what I am today.

Abstract

With ever-growing storage needs and drift towards very large relational storage settings, multi-relational data mining has become a prominent and pertinent field for discovering unique and interesting relational patterns. As a consequence, a whole suit of multi-relational data mining techniques is being developed. These techniques may either be extensions to the already existing single-table mining techniques or may be developed from scratch. For the traditionalists, single-table mining algorithms can be used to work on multi-relational settings by making inelegant and time consuming joins of all target relations. However, complex relational patterns cannot be expressed in a single-table format and thus, cannot be discovered.

This work presents a new multi-relational frequent pattern mining algorithm termed Multi-Relational Frequent Pattern Growth (MRFP Growth). MRFP Growth is capable of mining multiple relations, linked with referential integrity, for frequent patterns that satisfy a user specified support threshold. Empirical results on MRFP Growth performance and its comparison with the state-of-the-art multi-relational data mining algorithms like WARMR and Decentralized Apriori are discussed at length. MRFP Growth scores over the latter two techniques in number of patterns generated and speed.

The realm of multi-relational clustering is also explored in this thesis. A multi-Relational Item Clustering approach based on Hypergraphs (RICH) is proposed. Experimentally RICH combined with MRFP Growth proves to be a competitive approach for clustering multi-relational data. The performance and

quality of clusters generated by RICH are compared with other clustering algorithms. Finally, the thesis demonstrates the applied utility of the theoretical implications of the above mentioned algorithms in an application framework for auto-annotation of images in an image database. The system is called CoMMA which stands for Combining Multi-relational Multimedia for Associations.

Table of Contents

List of Tables

List of Figures

1. Introduction

A large volume of data is available to us today and its amount is continuously growing. Be it our decisions at the super market, the swipe of an access card, our search queries on engines like Google, browsing HTML pages on the world wide web or a visit to the hospital, every choice we make is recorded and adds to the already existing knowledge flood. Lying hidden in all this data is potentially useful information that is rarely made explicit or taken advantage of.

Knowledge discovery in databases (KDD) also referred as data mining, is the process of extracting useful hidden information from large volumes of raw data. Association rule mining is a KDD technique which searches for interesting relationships among items in a given dataset [1] [2] [3]. Typically association rule mining is used for extracting collections of statistically related data attributes from market basket data type transactions.

Clustering is another active topic in data mining research. It is the process of grouping objects into classes such that the objects within the class, called *cluster*, are similar to one another and are dissimilar to the objects in other clusters [4]. Clustering technique applies when there is no class to be predicted, thus it can be used for a descriptive data mining task where the objects have to be divided into natural groups. In machine learning, clustering is an example of unsupervised learning, which unlike classification does not depend upon pre-labeled training data.

While traditional data mining algorithms look for patterns in a single relation, multi-relational data mining (MRDM) is a multi-disciplinary field dealing with knowledge discovery from multiple tables (relations) of a relational database [5]. The

field aims at integrating results from existing fields such as inductive logic programming [6], KDD, machine learning and relational databases; producing new techniques for mining multi-relational data and applying them on real world problems.

Consider a database *D* that consists of *n* tables with one primary relation having a primary key *k* and *n-1* secondary relations referring to the key *k*. Then problem statement of this thesis is stated as:

> Developing an algorithm for finding frequent patterns across all *n* relations using multi-relational data mining and exploring the area of multi-relational clustering.

This thesis presents a new fast and scalable algorithm MRFP Growth for frequent pattern mining in multi-relational databases. MRFP Growth is an acronym for Multi-Relational Frequent Pattern Growth and as the name suggests, is an extension to the FP Growth algorithm [7] which discovers frequent patterns from a single relation without the generation of candidate itemsets.

The input given to MRFP Growth consists of tables and columns that need to be mined and the support counts and *cross support* to be considered as threshold for pattern generation. *Cross support* is a new measure used by MRFP Growth for measuring the frequency of an itemset in a database with multiple relations. The term is defined in section 3.2 of this thesis. MRFP Growth can mine together 'related' tables for frequent patterns. Two tables are 'related' if there exists referential integrity or some common attribute between the two tables (for example, parent and child tables) or if the two tables refer to the same table or have a common attribute between

them and the primary table (for example, two child tables referring to the same parent table).

The corner stone of MRFP Growth algorithm is ID set propagation. An ID set is a set of tuple IDs of the transactions in which the frequent itemset occurs. The idea behind ID set propagation is to be able to algorithmically join tuples in relations with minimum expense. This concept has the advantage that less frequent tuples in a table are filtered out and not considered for joining, thereby reducing the join cost and speeding up MRFP Growth. The algorithm also inherits a part of its celerity from FP Growth, as it needs only two scans per table.

MRFP Growth is not a sequential covering algorithm, as opposed to ILP approaches [7] [18]; it mines concurrently any number of tables, which may even belong to different databases, provided that they are 'joinable'. This is the first phase of MRFP Growth in mining tables for frequent patterns which provides considerable speedup to the algorithm.

The working of MRFP Growth starts with finding frequent patterns from individual tables. These patterns, along with their support counts and ID sets are placed in a temporary relation. This relation is then mined to realize frequent patterns across relations. A rule generation algorithm can then be applied to these frequent patterns to get the final multi-relational association rules.

To evaluate the applicability and robustness of MRFP Growth, a series of rigorous tests are performed on a variety of popular datasets PKDD and CoMMA, discussed in detail in the Experiments and Results chapter. MRFP Growth is also compared to the existing multi-relational association rule mining algorithms

WARMR [7] and Decentralized Apriori [8]. Runtime and number of frequent patterns generated with respect to different support values are compared for the given datasets.

This thesis further explores the area of multi-relational cluster discovery and proposes a new approach called multi-Relational Item Clustering using Hypergraphs (RICH), for clustering data. The technique of clustering using a hypergraph model was originally introduced in [9], though this work was done for high dimensional single table data and not multi-relational data. A hypergraph [10] is a generalized graph in which each hyperedge joins more than two vertices. The frequent patterns generated from MRFP Growth are converted into a hypergraph. Each item in a frequent itemset is represented as a vertex and the related items are connected using hyperedges. The graph is then partitioned, using a hypergraph partitioning algorithm to obtain clusters.

RICH performs item clustering and uses support to control the accuracy of the clusters generated. Empirical evaluation of RICH is done on both multi-relational and single-relation datasets. Performance and cluster quality of RICH is compared to that of CLUTO [11] clustering framework. CLUTO is single relational hypergraph-partitioning based clustering algorithm. Therefore, for comparison with CLUTO, only single relation, Mushroom and Congressional Voting datasets were used.

1.1 Knowledge Discovery in Databases

The process of data mining enables data exploration, analysis and visualization of very large databases. This non-trivial task of discovery needs to be automatic or semi-automatic. The information or patterns that are mined must be

valid, novel, meaningful (in that they can be used for some advantage) and ultimately understandable.

Data mining has a great potential to help businesses make proactive, knowledge-driven decisions, chalk out plans based on predicted future trends, improve customer relationship management, and help in achieving targeted marketing, besides other advancements. Some other areas of data mining application, to name a few, are astronomy, medicine, bioinformatics, government, law enforcement, and geophysics. Of late, data mining has helped companies reduce costs and customer attrition rate, improving sales effectiveness and profits.

The process of knowledge discovery in databases consists of an iterative sequence of the following steps [12]:

Problem definition, in which the goals of the knowledge discovery project must be identified and must be verified as actionable.

Data preprocessing which includes data collection from various sources, data cleaning to remove noise and inconsistent data, data integration for combining data from multiple sources, data selection to retrieve data relevant to the analysis task and data transformation for consolidation of data into forms appropriate for using.

Data mining is an essential process where intelligent methods are applied in order to extract data patterns. It searches for patterns of interest in a particular representational form such as rules, clusters, classification rules/trees and so forth. This step may interact with the knowledge base and the interesting patterns found are presented to the user and possibly stored as new knowledge in the database.

Post data mining processes include pattern evaluation to identify the truly interesting patterns representing knowledge based on some interesting measures, model deployment and maintenance and representation of knowledge.

There are various types of data mining tasks like association rule mining, classification, clustering, outlier detection, trend analysis, deviation analysis, and similarity analysis. *Association analysis* is the discovery of association rules showing attribute-value conditions that occur frequently together in a given set of data. Association analysis is widely used for market basket or transaction data analysis. *Classification* is the process of finding a set of models that describe and distinguish data classes or concepts, for the purpose of being able to use the model to predict the class of objects whose class label is unknown. This kind of learning technique is known as supervised learning. *Clustering* unlike classification analyzes data objects without consulting a known class label. The objects are clustered into groups based on the principle of maximizing the intra-class similarity and minimizing the interclass similarity.

1.2 Multi-Relational Data Mining

The idea of mining from multiple tables is not a new one. It is being studied extensively in the field of Inductive Logic Programming [6]. However these approaches are mostly based on data stored as Prolog programs, and little attention has been given to data stored in relational database.

Efficiency and scalability have been of major concern in the data mining field. They are even more so when the focus is on multi-relational data mining. Joining data from multiple relations into a single table requires much thought and

effort and can lead to loss of information or excessive redundancy [13]. Moreover the computationally expensive process of relational joins resulting in a single large relation can have a huge impact with respect to memory management. From a practical point of view, significant speedup can be obtained if the relations are kept as it is so that the applicable subset of tuples fit in main memory. Thus, squeezing data from multiple tables for analysis by classical data mining techniques is unacceptable or to say the least inappropriate. Multi-Relational Data Mining on the other hand aims to take advantage of the semantic information carried by semantic links while mining these relations. MRDM can analyze data from a multi-relational database, without the need of transferring the data into a single table first. Thus the relations mined can reside in a relational or deductive database. Using MRDM it is often also possible to take into account background knowledge, which often corresponds to views in the database.

Present MRDM approaches consider all of the main data mining tasks, including association analysis, classification, clustering, learning probabilistic models and regression. The pattern languages used by single-table data mining approaches for these data mining tasks have been extended to the multiple-table case. Relational pattern languages now include relational association rules, relational classification rules, relational decision trees, and probabilistic relational models, among others. Relational patterns are typically expressed in subsets of first-order logic (also called predicate or relational logic) [5]. MRDM algorithms have been developed to mine for patterns expressed in relational pattern languages.

A pattern language typically contains a very large number of possible patterns even in the single table case. For relational pattern languages, the number of possible patterns is even larger and it becomes necessary to limit the space of possible patterns by providing more explicit constraints. These typically specify what relations should be involved in the patterns, how relations can be interconnected, and what other syntactic constraints the patterns have to obey [5]. The explicit specification of the pattern language is known as declarative bias [14].

MRDM methods have been successfully applied across many application areas, ranging from the analysis of business data, through bioinformatics (including the analysis of complete genomes) and pharmacology (drug design) to Web mining (information extraction from text and Web sources).

1.3 Association Rule Mining

Association rule mining searches for interesting relationships among items in the given dataset [1]. Typically association rule mining is used for extracting collections of statistically related data attributes from market basket data type transactions. Table 1 shows a set of morning breakfast-items purchased by some customers of a grocery store.

Table 1: A Table containing Market Basket Data

TID	Item
1	Bread Cereals Eggs
2	Milk Bread
3	Cereals Bread Sugar
4	Milk Bread Juice
5	Milk Bread Cereals

Market basket analysis of such customer transactions at a store can assist in store layout, marketing or advertising strategies, client offers and catalog designing. Rules mined from such transactions are of the form Cereals → Bread. Some other applications of rule mining are finding irregularities in data, finding frequent, sequential, structural, or periodic patterns in data, correlation and causality analysis.

The interestingness of rules mined is measured in terms of *support* and *confidence* of a rule. The rule A → B holds in a transaction set D with *support* s, where s is the percentage of transactions in D that contain A U B, i.e. both A and B. The rule A → B has *confidence* c in the transaction set D if c is the percentage of transactions in D containing A that also contain B. A rule is considered to be interesting if it satisfies both minimum support and minimum confidence. Rules obtained from table 1 for a minimum support=20% and minimum confidence=50% are given in Table 2.

Table 2: Rules mined from Table 1 with Support(s) = 20% and Confidence(c) = 50%

Rules
Cereal → Bread (s= 60%, c=100%)
Milk → Bread (s=60%, c=100%)
Bread → Cereal (s=60%, c=60%)
Bread → Milk (s=60%, c=60%)

Association rule mining is a popular research area in the field of knowledge discovery and several algorithms have been developed to this end. All of these algorithms have their own advantages and disadvantages and have been compared in [15]. One of the most widely used algorithms for association rule mining is the

Apriori algorithm [2]. The algorithm exploits the *anti-monotone* property which states that for a k-itemset to be frequent all (k-1) subsets of this itemset also have to be frequent. Though the algorithm reduces the computational cost of generating the itemsets, the computational cost is still high when the number of 1-frequent itemsets is sufficiently high, which in turn translates into a high cost for generating 2-frequent itemsets. The FP-Growth algorithm [3] was proposed to overcome this problem. The algorithm creates a compact tree-structure called the FP-Tree that represents frequent patterns and mines the FP-Tree to get the frequent patterns. It solves the multi-scan problem and improves itemset generation.

1.4 Clustering

Clustering is another active topic in data mining research. It is the process of grouping objects into classes such that the objects within the class, called *cluster*, are similar to one another and are dissimilar to the objects in other clusters. Clustering technique applies when there is no class to be predicted, thus it can be used for a descriptive data mining task where the objects have to be divided into natural groups. In machine learning, clustering is an example of unsupervised learning, which unlike classification does not depend upon training class-labeled training data.

Clustering is being used in business for discovering distinct customer groups so as to provide customized solutions to them. In astronomy, clustering is used to find groups of similar stars and galaxies. Generating plant and animal taxonomy and categorizing genes is a biological application area of clustering. WWW, earthquake studies, demographic studies, are a few other application areas of clustering.

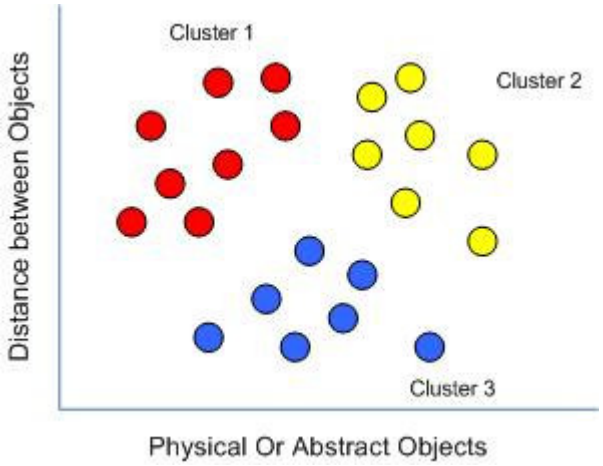

Figure 1: A Clustering Task.

Figure 1 depicts an example of a clustering task. Clusters generated from a clustering algorithm can either be exclusive (non-overlapping), overlapping, hierarchical or probabilistic. Based on the type of clusters formed, there is a gamut of clustering methods; most general of them being partitioning, hierarchical, density-based and grid-based methods [4]. These methods can further be classified into numerical or categorical methods, based on the type of data to be clustered.

1.5 Roadmap

The next chapter discusses the existing multi-relational association rule mining and clustering algorithms, WARMR, Decentralized Apriori and RIBL. Chapter 3 describes the MRFP Growth algorithm in detail. It starts out with a concise explanation of how FP Growth works, and then gives the basic definitions, core concepts and the algorithms used in MRFP Growth, ending with a detailed example showing how MRFP Growth mines multi-relational data for frequent patterns. Chapter 4 explores the topic of multi-relational clustering and talks about the RICH approach to clustering multi-relational data. Chapter 5 introduces CoMMA to the

readers. The technique used for auto annotation of images, the results and the evaluation metrics for the results have been detailed out. Chapter 6 gives the experiments and results of running MRFP Growth and RICH on different datasets. Chapter 7 concludes this thesis.

2. WARMR, Decentralized Apriori, RIBL

This section talks about the existing multi-relational association rule mining and clustering algorithms. Several multi-relational methods to analyze data have been developed by the Inductive Logic Programming community over the recent years. The ILP approaches achieve a good accuracy in data analysis. However they are usually not scalable with respect to the number of relations in the database and the number of attributes in the database. Therefore these approaches are inefficient for databases with complex schemas. Another drawback of the ILP approaches is that they all need the data in the form of prolog tables. None the less, these are state-of-the-art methods and are discussed here so as to get a better understanding of how they work and how MRFP Growth can be made to score on them.

2.1 WARMR

WARMR [7] is a relational data mining system for discovering frequent patterns. It is an extension to the APRIORI algorithm [2] that discovers frequent patterns from a given dataset by doing a breadth first search through the lattice of itemsets. WARMR takes as input a database *D*, a frequency threshold *minfreq,* and the declarative language bias *L*.

WARMR searches through a lattice of Datalog[1] queries for queries that are frequent in the given database *D*. A query is a set of atoms where all variables are existentially quantified and the set is ordered. The free variables in a query are bound

[1] Datalog is a database query language that syntactically is a subset of Prolog (en.wikipedia.org/wiki/Datalog).

by a special purpose key predicate. The relation of the key, k, and the query is illustrated in the following Horn clause:

$$k(\mathrm{X}) \leftarrow account(\mathrm{X}, \mathrm{Y}, \mathrm{credit}), loan(\mathrm{X}, \mathrm{monthly}) \quad (1)$$

In this clause the predicates *account* and *loan* are relations that hold account information and loan information respectively of the clients of a bank. The support of the query is formalized using the key and is defined to be the number of variable bindings for which the key predicate can be proved. In the given example the support of $k(\mathrm{X}) \leftarrow account(\mathrm{X}, \mathrm{Y}, \mathrm{credit}), loan(\mathrm{X}, \mathrm{monthly})$ is the number of variable bindings of X for which $k(\mathrm{X})$ can be proved given the Horn clause in rule (1) and a knowledge base defined in PROLOG.

In analogy to itemsets in APRIORI, given a simple query Q1, a more complex query Q2 can be generated from it. However, unlike itemsets the definition of the search space is not straightforward for atom sets. Apart from the choice of predicate, there are also many possibilities for the usage of variables in the query. To define the *bias* of the search space WARMR uses a refinement operator based on *mode declarations*. Every mode declaration prescribes the way in which a predicate can be added to a query. For example, following is the example of a mode in the declarative language bias *L*:

```
warmode_key(account_number(+)).
warmode(account(+, -, credit)).
warmode(account(+,-,debit)).
warmode(loan(+,monthy)).
warmode(loan(+,yearly)).
```

The above bias *L* shows that *account_number*(X) as the key atom and the gives the input-output modes for the relations account and loan. Input-output modes specify whether a variable of an atom in a query has to (+), must not (-), may, but need not (+-) appear earlier in the query. For example, the second mode in the bias states that the predicate *account* may be added to a query when the first parameter is bound to an existing variable, the second parameter introduces a new variable and the last parameter is bound to the constant credit.

Considering the declarative bias given above, WARMR starts with queries at level 1 with the query ? – *account_number*(X). At level 2, the literals *account*(+,-,credit) and *account*(+,-,debit) can be added to this query. The following candidate queries yield from the level 1 query: ?-*account_number*(X), *account*(X,Y,credit) or ?-*account_number*(X), *account*(X,Y,debit) or ?-*account_number*(X), *loan*(X,monthly) or ?-*account_number*(X), *loan*(X,yearly). Taking first query from the queries at level to the following literals can be added to obtain queries at level 3: *account*(Y,Z,credit), *loan*(Y,monthly), *loan*(Y, yearly).

APRIORI uses the anti-monotone property (as discusses in Chapter 1) for itemsets, but for WARMR not all subqueries of a frequent query need be frequent queries. Consider the query ?-*account_number*(X), *account*(X,Y,credit), loan(Y, yearly) to be a frequent query. Then the subquery ?-account_number(X), loan(Y,yearly) is not allowed as it violates the declarative bias constraint that the first argument of *loan* has to appear earlier in the query. Thus, the usage of atoms instead of items turns it more difficult to create an efficient APRIORI-like algorithm: it is no longer reasonable to use the subset relation to prune the candidates for frequent

queries. Instead WARMR keeps a list of infrequent queries and checks whether the generated candidates can be *θ-subsumed* by a query in this list.

A substitution $\theta = \{V_1 / t_1 , \ldots , V_n / t_n\}$ is an assignment of terms t_i to variables V_i. Applying a substitution to a term atom or clause A yields the instantiated term, atom, or clause $A\theta$ where all occurrences of variable V_i are simultaneously replaced by the term t_i [5]. Let A and B be two atom sets. Then set A θ-subsumes atom set B, denoted by $A \leq B$, if there is a substitution θ such that $A\theta \subseteq B$ [16].

A major problem of WARMR is that it heavily depends on a good implementation of subsumption. This is prohibitive as θ-subsumption is an NP-complete problem [17].

2.2 Decentralized Apriori

Jensen and Soparkar [8] propose a frequent itemset mining algorithm for decentralized data. It exploits the inter-table foreign key relationships to obtain decentralized algorithms that execute concurrently on separate tables and thereafter merge the results. The decentralized apriori algorithm is an extension to the existing APRIORI [2] algorithm and can be applied to datasets that have a star schema. The decentralized approach is a two phase startegey:

- Find the frequent itemsets on individual tables separately, and then
- Merge results from individual tables by using foreign key relationships.

The algorithm for Decentralized Apriori is discussed below. Consider n primary tables which contains the data to be mined and one central relationship table (the fact table): T_{1n}. Each table $T_t(id_t; a_{t1}, \ldots, a_{tn})$ has a primary key id_t, and T_{1n} (id_1, id_2, …..,

id_n) has id_t as foreign key to table T_t. The task of finding frequent itemsets can be described as follows:

Phase I:

1. Count the occurrences of each value for id_t
2. Store each value in a vector v_t for each V_t such that the number of elements in T_i equals the number of rows of T_t *i.e.,* $|V_t| = |rows(T_t)|$
3. Apply the traditional Apriori algorithm on each table. The item are counted such that the support of an item in i^{th} row is incremented by number of occurrences of id_t for row i in T_{1n}. This results in n sets of frequent itemsets, l_t from table T_t. The itemsets can be of length 1 up to m_t (the number of attributes in table T_t).

Phase II:

1. Count itemsets across primary tables using the relationship table.
2. Generate candidates from the n primary tables using an n-dimensional count array, where dimension t corresponds to elements of the set together with the empty set.
3. Compute the joined table without materialization. For each row r in T, consider the corresponding attributes that come from each table T, and identify the subset of itemsets.
4. Each position in the n-dimensional array is incremented by one whenever an element I_T is formed by concatenating an element of i_1, an element of i_2, . . . , and an element of in (i.e., i_1, i_2, ……., i_n). I_T is analogous to an itemset contained in table T, such that its items belong to more than one primary table.

5. The resultant n-dimensional array contains the support for all the candidate itemsets.

Decentralized Apriori is a much better way to mine decentralized tables than by first joining and then mining them. However its scalability is limited as it uses a matrix structure during its second phase of execution. Moreover, as Decentralized Apriori and WARMR both build up on APRIORI, which doesn't use any specific data structure; they both need several table scans, which leads to heavy disk acceses, i.e. I/O time. The use of an efficient data structure for mining itemsets can remedy this problem. In addition, in some cases APRIORI may generate a huge number of candidate itemsets, while the number of frequent itemsets actually found is very small.

2.3 RIBL

Most distance-based clustering techniques use either, the Euclidean distance, the Manhattan distance or the Minkowski distance, which is a generalization of both the former distances for clustering data. Such distance measures however cannot be directly applied to the multi-relational scenario. In the Relational Instance Based Learning, RIBL [18] system, a new distance measure called RIBL has been introduced which is applicable on relational data.

Traditionally distance between two tuples $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ in a single relation is calculated as

$$\text{distance}(x, y) = \sum_{i=1}^{n} difference(x_i, y_i) / n \qquad (2)$$

where the difference between attribute values is defined as

$$\text{difference}(x_i, y_i) = \begin{cases} |\, x_i - y_i \,| & \text{if continuous} \\ 0 & \text{if discrete and } x_i = y_i \\ 1 & \text{otherwise} \end{cases} \quad (3)$$

In the RIBL distance measure, distance between two tuples is calculated by considering the distance between their properties first (at depth 0). Then the distance between the objects immediately related to the two given objects is calculated (at depth 1) and so on, till a user specified depth is reached [5]. For example, consider the following prolog ground atoms given for bank data:

account(acct1, John Paul, checking).

loan(acct1, homeloan, 350).

transaction(acct1, IL, 150, 230).

transaction(acct1, CA, 50, 180).

branch(IL, 5, Chicago).

branch(CA, 6, Palo Alto).

Here *account* is the target relation with a primary key account id, acct1, and other attributes like account holder's name and the type of account. The other relations: *loan* and *transaction* refer to the account relation's primary key account id. The *transaction* relation refers to the *branch* relation's primary key. Given another *account* tuple say *account*(acct2, Marie Sue, credit), the RIBL measure starts off computing distance between first account names and account types. Then moves on to the related tuples at depth=1. So, finds distance between *loan* tuples referring to acct1 and acct2 and then distance between *transaction* tuples referring to acct1 and acct2. Then the measure finds the distance between the related tuples at depth=2, and

so on. The case defined by *account*(acct1, John Paul, checking) with respect to the above background knowledge and for a depth 2 shown in Figure 2.

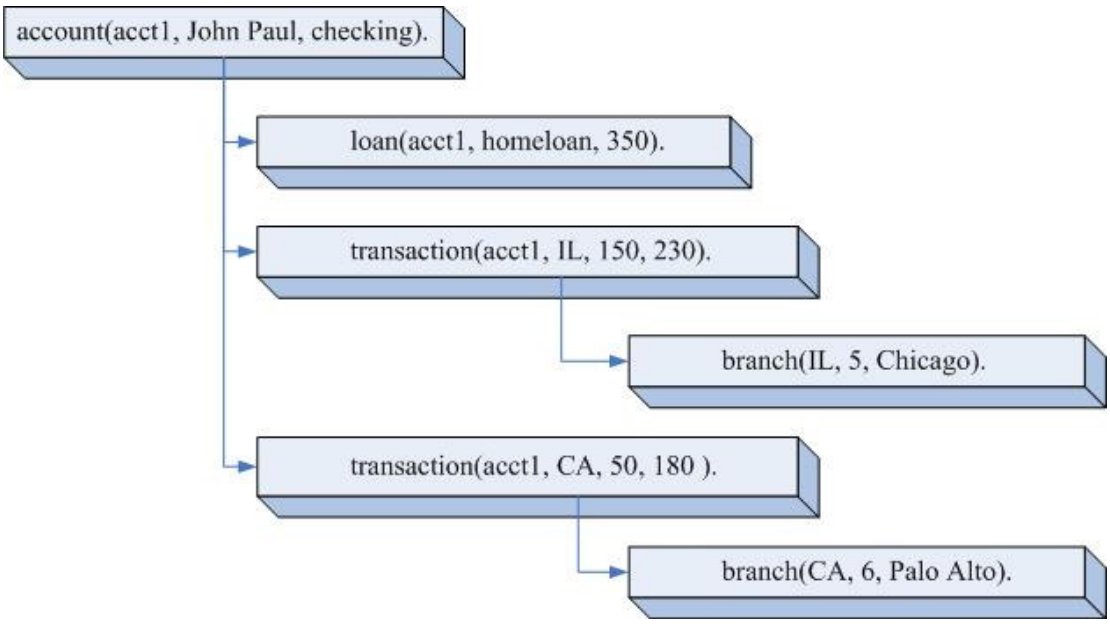

Figure 2: The case defined by *account*(acct, John Paul, checking) for depth=2.

This distance measure has been used by RDBC [19] that performs agglomerative hierarchical clustering and FORC [19] adapts the k-means clustering approach to work on relational data by using the RIBL distance measure. An advantage of this approach is that it considers the related objects when computing distances. The disadvantage is that it is too computationally intensive and expensive because of the huge number of related objects.

3. Multi-relation Frequent Itemset Mining

MRFP Growth algorithm is an algorithm for multi-relational frequent pattern generation. It is an extension to the popular FP Growth algorithm [3] that is applicable on single-relational setting. In this chapter, FP Growth has been discussed initially, to give background knowledge of MRFP Growth to the reader. Then the working of MRFP Growth has been explained in detail starting with the basic definitions, the core fundamentals and then the MRFP Growth algorithm followed by an example.

3.1 FP Growth

FP Growth is a frequent pattern mining algorithm which mines an FP-Tree, a frequent pattern tree, for frequent itemsets without candidate pattern generation. An FP-Tree is an extended version of prefix tree which stores frequent pattern information in a compressed format. Using such an efficient data structure for pattern storage drastically reduces the huge number of database scans (I/O time), which is needed otherwise. FP Growth needs only two scans of database to build the FP-Tree. It then uses a divide and conquer partitioning strategy to mine the FP-Tree generated. As no candidate itemsets are generated, no time is spent on unnecessarily matching patterns and generating candidates which may later prove to be infrequent.

MFRP Growth starts by first making the FP-Tree and then mines it for frequent patterns. The first scan generates all the 1-frequent itemsets. FP-Tree generation can be split into two phases. In the first phase, the items appearing in the dataset are enumerated. All the items that have a support less than the threshold are

weeded out. The remaining itemsets are organized in a table called the header table and are sorted by frequency. Pointers to the first occurrence of the items in the dataset are also stored in order to maintain reference for all other occurrences of the item. The second phase starts with another I/O scan of the database. Each transaction is read again and only those items that occur in the header table are inserted into the FP-Tree in the order of descending frequencies. Thus transaction by transaction, items are added to the FP-Tree.

Table 3: A Sample Relation

TID	Item
1	A, B, C, D
2	E, G, H
3	A, B
4	C, D, A
5	G, E, C
6	F, H, G
7	D, E, B

Table 3 shows a sample relation with a set of transactions that consisting of different items A, B, C, D, E, F, G, H, K, and L. Assume the support count to be 2, that is only items with frequency equal to 2 or above are considered to be frequent. This database (Table 3) is scanned and the frequency of occurrence of each item is counted. Then the item below the support, in this case F, is removed to get the header table shown in Figure 3.

In the second database (Table 3) scan, an FP-Tree is constructed on this sample relation as follows: read the items in a transaction in a vector. Sort the vector according to the order given in header table. Check if the first item in the sorted

vector exists as one of the children of the root node. If it exists then increment its support. If not, then add the current item as a child of the root node and set the support to 1.

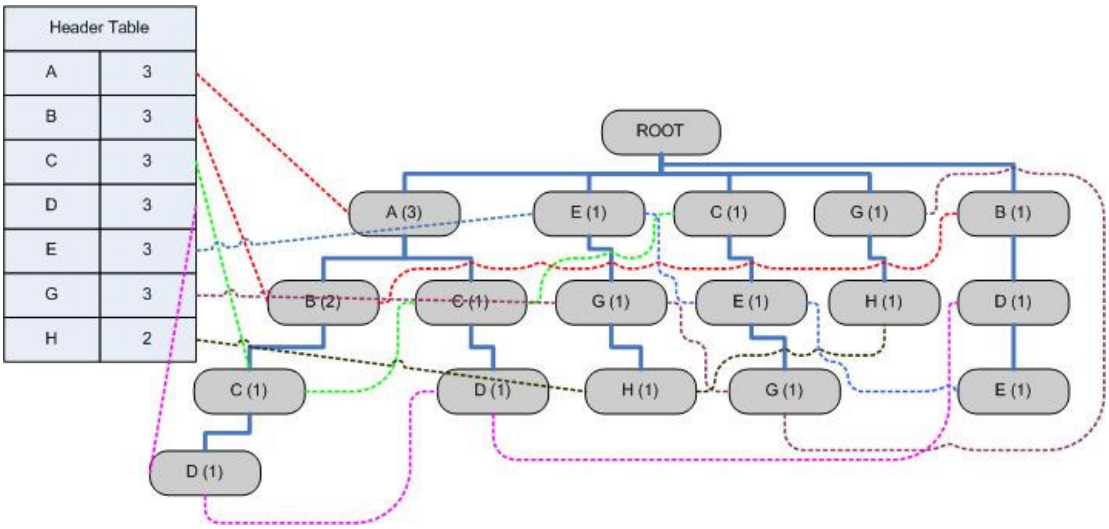

Figure 3: An FP-Tree for the sample relation in Table1 for support count ≥ 2

Repeat the same process for the second item in the sorted vector this time considering the current node as the root. Once all items in the sorted vector are added to the tree, read in another transaction and add its items in the same way to the tree. Whenever an item is added to the FP-tree a link is added to it its occurrence in the header table if it's the very first occurrence of the item in the tree, or a link is made to its previous occurrence. The process continues till the whole FP-Tree is generated. The complete FP-Tree for the sample relation in Table 3 is given in Figure 3.

Mining the FP-Tree is summarized as follows. Start the mining process from the last item in the header table, moving upwards. For each frequent length pattern (as an initial suffix pattern) construct its conditional pattern base, then construct its conditional FP-Tree and perform mining recursively on each such tree. A conditional FP-Tree is an FP-Tree built on a conditional pattern base, which is a subpattern-base

under the condition of existence of a suffix pattern. The pattern growth is achieved by concatenation of the suffix pattern with the frequent patterns generated from conditional FP-Tree.

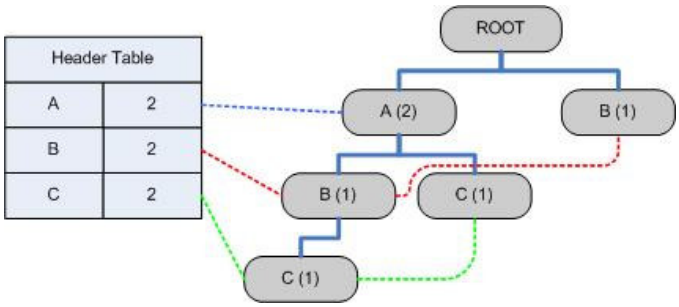

Figure 4: Conditional FP Tree for suffix pattern D

Figure 4 shows the conditional FP-Tree for item D. This tree is again mined using FP Growth and this recursive mining continues till either a single path conditional base is left in the tree or only one item is left. When a single path is left, combination of all items in the path appended to the suffix pattern becomes the set of frequent patterns. If only one item is left, then the item union the suffix pattern is the frequent pattern. Each, item, its conditional pattern base, conditional FP-Tree and frequent patterns generated are shown in Table 4.

Table 4: Frequent Patterns generated from given tree

Item	Conditional Pattern Base	Conditional FP-Tree	Frequent Patterns
H	{(E G:1), (G:1)}	<G:2>	H G:2
G	{(E:1), (E C:1)}	<E:2>	G E:2
E	{(C:1), (B:1)}	<>	-
D	{(A B C:1), (A C:1), (B:7)}	<A:1 B:1 C:1> <A:1 C:1> <B:1>	D C:2, D B:2, D A:2, D A C:2
C	{(A B:1), (A:3)}	<A:2>	C A:2
B	{(A:2)}	<B A:2>	B A:2
A	{}	<>	-

More detail on construction of FP-Trees can be found in [3]. FP Growth can find both long and short frequent patterns efficiently. Using the divide and conquer strategy it splits the problem of finding longer frequent patterns by looking for shorter ones recursively and then concatenating the suffix, thereby substantially reducing the search costs.

3.2 Basic Definitions

Consider a database *D* that consists of *n* tables with one primary relation having a primary key *k* and *n-1* secondary relations referring to the key k. Each relation may have one primary key and several foreign keys. MRFP Growth considers the following types of joins:

1. Join between a primary key *k* and the foreign keys referring to *k*.
2. Join between foreign keys referring to the same primary key *k*.

Other possible joins are not considered strong relationships between entities and therefore not used for joining relations by MRFP Growth. Following are some definitions for the terms that are used in the MR FP-Growth algorithm.

1. *Primary Relation*: A relation that has a primary key *k*. Example: Table 3 with TID as the P_k (Primary Key).
2. *Secondary Relation*: A relation that has attribute(s) which refer to the primary key *k* of the primary relation. Example: Table 6 and Table 7 have TID as the F_k (Foreign Key).
3. *ID*: The primary key value for a transaction. Example: *ID* for item H in Table 3 is: *ID*[H] = {2, 6}.

4. *ID set*: The set of IDs in which a frequent pattern occurs. Let A_i be an item in a frequent pattern P. Then *ID set* of $P = \left\{ \bigcap_{i=1}^{k} ID[A_i] \right\}$.

5. *MR FP-Tree*: A frequent pattern tree (extension to a prefix tree) which holds frequent patterns along with their support count and the ID set in a compressed format. The ID set helps in determining the link information between the different relations.

6. *Cross Support* χ : A frequent pattern *P* holds in a database *D* with a cross support χ if *P*'s frequency of occurrence in D is χ. Formally, cross support for *P* is χ if *P*'s ID set is repeated χ times in different frequent patterns in any relation in the database *D*. For example, a frequent pattern A, B, C, D, E, G {1,4} is said to have a cross support χ = 3 if the patterns A, B {1,4} has support = 5, C, D {1,4} has support = 3 and E, G {1,4} has support = 10 are frequent in the relations R_1, R_2 and R_3 of database *D* respectively.

Besides these terms, other terms associated with association rule mining such as frequent itemsets and support values are used frequently in the following sections. These terms were explained in detail in Section 1.3. To recap, a set of items is called an *itemset*. An itemset is said to be a *frequent itemset* if it exceeds a user specified threshold called support. A frequent pattern is said to have a *support* s, if the pattern appears in s% of the transactions in a relation. Please note that the terms frequent itemset and frequent pattern will be used interchangeably in the text.

3.3 Core Fundamentals

The essence of a true multi-relational algorithm is to be able to mine novel patterns in a single relation and also find patterns that span related relations. MRFP Growth uses a two step approach for finding frequent itemsets:

1. It finds frequent itemsets in each relation, both primary and secondary.
2. It mines the frequent itemsets found in individual relations, to find itemsets across relations.

Step1 mines each relation individually therefore the mining process of a single relation is independent of other relations. In effect, these relations can be mined concurrently to reduce execution time, which makes MPFP Growth a fast and scalable algorithm. Fast, as the relations can be *parallel processed* which saves considerable amount of time. *Scalable* as the entire database is not considered in a single run, this may have been the case if the relations were joined to make a single huge relation. The fragmented approach ensures that each relation is mined without the out of memory error.

The independent mining of relations in step 1 is responsible for another unique feature of MRFP Growth, the ability to *mine and retain individual relation frequent patterns*. This feature is missing in any of the multi-relational ILP [7] [18] approaches and other sequential covering algorithms that start from a target relation and work their way through related relations. Such algorithms consider only those transactions for mining frequent patterns from secondary relations that have been found to be frequent in the target relation. As a result, several transactions that may have frequent patterns are not considered for mining which leads to loss of important

information. MRFP Growth on the other mines the frequent patterns from individual relations, independent of each other and retains them if they have a support count greater than the cross support.

MRFP Growth takes one primary relation and any number of secondary relations as input for mining. Let us call a single input to MRFP Growth as an input set, which consist of a single primary relation and optional multiple secondary relations. If no secondary relation is specified, then MRFP Growth behaves like the FP Growth algorithm, and does not execute the second step at all. Thus FP Growth is a special case of MRFP Growth where number of relations to be mined = 1. If there are more than one input sets that need to be mined, then MRFP Growth can be used incrementally to mine one set at a time, and finally merge the patterns found by all the sets and mine these patterns.

Besides the relations, specific attributes in each relation that need to be considered for frequent pattern generation can be given as input to MRFP Growth. This allows *enhanced selectivity* in terms of attributes to be used for mining frequent itemsets. The user can therefore exclude those attributes from pattern generation, which need not be considered or may not produce intelligent patterns.

The corner stone of MRFP Growth algorithm is ID set propagation. The idea behind ID set propagation is to algorithmically join frequent tuples in 'related' relations with minimum expense. This concept has the advantage that less frequent tuples in a relation are filtered out and not considered for joining, thereby reducing the join cost and speeding up MRFP Growth. The algorithm also inherits a part of its celerity from FP Growth, as it needs only two table scans per relation.

3.4 MRFP Growth Algorithm

The Multi-relational FP-Growth algorithm is given below. MRFP_Growth uses MRFP_Tree and MRFP_Mine algorithms for creating MRFP Trees and mining them. It also uses ID_Item_Mapping procedure for mapping IDs to actual items in the final phase.

3.4.1 MRFP_Growth Algorithm

Algorithm: MRFP Growth that mines multiple relations for frequent patterns.

Input: A primary relation P, its primary key Pk, n secondary relations Sn, their attributes and support counts and the cross support χ.

Output: The complete set of multi-relational frequent patterns.

Method:

1. For each secondary relation S_n do:
    i. Generate MRFP-Tree for the items in the relation using the *MRFP_Tree* algorithm. Keep track of the ID for each frequent item.
    ii. Mine the MRFP-tree for frequent patterns using *MRFP_Mine* algorithm. Also, note the ID sets of the frequent patterns.
2. Make an MRFP-tree using *MRFP_Mine* for the ID in the ID sets of all the frequent patterns generated for all the secondary relations.
3. Mine the final MRFP-tree, using MRFP_Mine algorithm. For the frequent patterns of ID, get the actual patterns associated with these IDs using ID_item_Mapping procedure and effectively get the frequent patterns across the relations.

The first phase of the algorithm involves running the MRFP_Tree algorithm separately on all the relations. Each node in the tree not only keeps track of its support but also keeps track of the indices (ID) in the dataset where its item occurs. The algorithm for MRFP Tree generation is given below:

3.4.2 MRFP_Tree Algorithm

Algorithm: MRFP_Tree creates an MRFP-Tree for the given relation.

Inputs: A relation S, a set of attributes A of the relation and a support count s (or cross support χ for final MRFP-Tree).

Output: An MRFP-Tree for the given relation

Method:

1. Scan the relation, S once while counting each item in the each attribute A_i of the given attribute set A. Make a list of all 1-frequent items that are greater than s (or χ). The list (header table) contains items and support values in descending order of support.
2. Create an MRFP-Tree node called root and label is as null. Scan the relation once again. Read in each transaction T in S and do as follows:
    a. Select and sort the frequent items in T according to the order given in the header table. Let p be an item in the sorted list P of frequent items in T.
    b. Call insert_item(P, support_count, ID). The method insert_item, looks for p_1 in Root node's child list.
        i. If p_1 is not found then it adds a new node N with item p_1, support as the node's count, ID as the nodes ID set and null as the node's

follower. If N is the very first node with item p_1 in the tree, then a link from the header table to N is added. But, if p_1 has occurred previously in the tree, then the link from the header node is followed till last node is found; for this last node, the follower is set to N.

ii. If p_1, node N, is found as one of Root's children, then it increments N's support count by one and performs a logical OR operation of N's ID set with the ID of p_1.

iii. It then repeats the search for p_2, starting from p_1's node this time. And continues with the same steps b(i), b(ii) and b(iii) till all items in P have been processed and the pattern is added to the tree.

There are different ways by which ID sets can be stored in each node of the tree. In the approach mentioned above, the MRFP-Tree nodes store ID sets in the form of bitmaps. Alternatively, the ID sets can be stored as strings. Once each tree is made, it is mined for frequent patterns using the MRFP_Mine algorithm given below.

3.4.3 MRFP_Mine Algorithm

Algorithm: MRFP_Mine that mines a given MRFP-Tree for frequent patterns

Inputs: An MRFP-Tree T of a relation, the suffix pattern α, which is initially null.

Output: All frequent patterns generated from S's tree.

Method:

1. If T contains a single path P then

a. For each combination (denoted as β) of the nodes in the path P generate pattern β U α with support = minimum support of nodes in β and ID set = {(ID set of β) ∩ (ID set of α)}. Place the frequent pattern found, along with a unique identification number, the support and the ID set information in the table called frequent_patterns (or call ID_Item_Mapping procedure with the ID set information).

2. Otherwise for each a_i in the header of T do as follows:

   a. Generate pattern $\beta = a_i$ U α with support = a_i.support and ID set = a_i.ID set.

   b. Construct β's conditional pattern base and then β's conditional MRFP-Tree $T_{\beta.}$

   c. If $T_\beta \neq$ null then call MRFP_Mine(T_β, β).

Phase two finally joins the relations by using the table that was filled up with the frequent patterns mined from MRFP-Trees in Phase 1, to make an MRFP-tree on the ID set. Each ID in an ID set is considered to be an item and each unique number identifying a frequent pattern is treated as an ID. This final tree is made using the MRFP_Tree algorithm along with the given cross support. Once the tree is made, it is mined for patterns using the algorithm MRFP_Mine. However, this time the frequent patterns found consist of IDs and are mapped to real items using the ID_Item_Mapping procedure described below.

3.4.4 ID_Item_Mapping Procedure

Procedure: ID_Item_Mapping finds frequent patterns across relations.

Inputs: ID set of the ID frequent pattern found from frequent_patterns table.

Output: The set of actual frequent patterns for the given ID set.

Method:

1. Find all records in the frequent_patterns table with ID in the given ID set.
2. Make a new pattern P.
3. For each transaction T in records found do:
    a. P=P U T, P.support = minimum T.support found, and P.ID set = P.ID set ∩ T.ID set.
    b. Add T to the rules table with T.support as support and T.ID set as ID set.

4. Add P to the rules table.

In this way finally rules across different relations are generated. To get a better understanding of the algorithms explained in this section, the reader is advised to go through the following example section.

3.5 An MRFP Growth Example

Table 5: Primary Relation

TID	Item
1	A, B, C, D
2	E, G, H
3	A, B
4	C, D, A
5	G, E, C
6	F, H, G
7	D, E, B

Table 6: Secondary Relation 1		Table 7: Secondary Relation 2	
TID	**Items**	**TID**	**Items**
1	α, β, γ	1	Aa Cc
1	β, δ	2	Bb Dd Aa
2	δ, θ, η	2	Bb Ee
3	α, θ	3	Cc Ff
4	β, η, γ	4	Hh Dd Gg
5	α, δ	6	Gg Ff
5	α, η, γ	6	Aa Cc
7	β, η	7	Ee

A running example in this section will use the Table 5, Table 6 and Table 7 as the primary relation and the two secondary relations respectively. TID is the primary key in Table 5 and foreign key in Table 6 and Table 7. There is a one-to-many relationship between the primary and secondary relations. Running MRFP_Tree on Table 5 yields the following MRFP-Tree (Figure 5). The support count for is 2, that is all items having frequency of occurrence greater than or equal to 2 are considered frequent.

Each node stores ID set for the corresponding node item. The ID set is shown in braces and frequency of occurrence is indicated in round brackets. In the node B (2) {1,3}, B is the node's item, 2 is its frequency of occurrence and {1,3} indicates the ID set where B occurs. A link from the header table to this node shows that this is the first occurrence of item B's node in the tree. The node B (1) {7} is the follower of the node B (2) {1,3}.

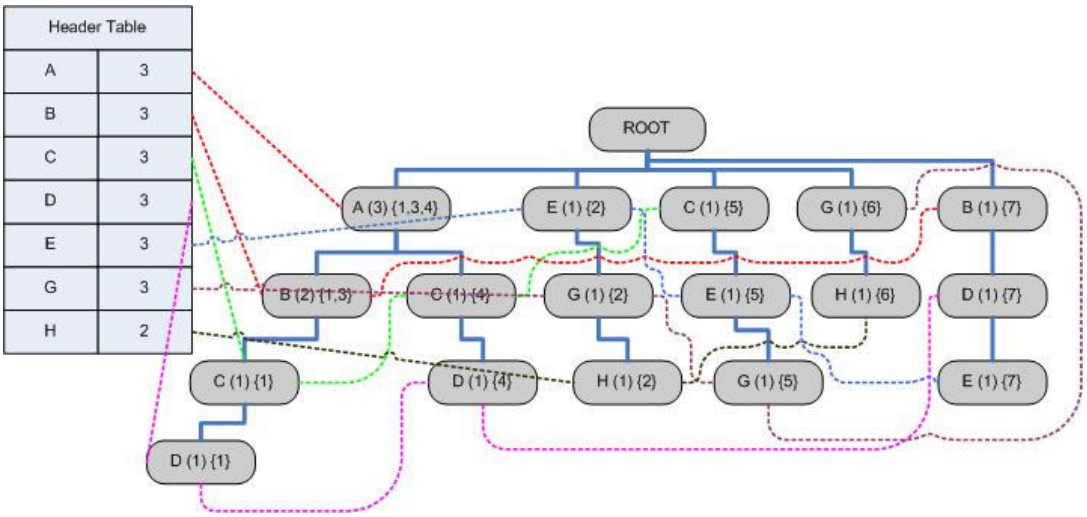

Figure 5: MRFP-Tree for the Primary Relation

Mining this tree yields the following frequent patterns (Table 8).

Table 8: Frequent Patterns table after mining MRFP-Tree in Figure 5

ID	ID set	Frequent Patterns	Support
1	2 6	H G	2
2	2 5	G E	2
3	1 4	D C	2
4	1 4	D C A	2
5	1 7	D B	2
6	1 4	D A	2
7	1 4	C A	2
8	1 3	B A	2

Once the primary relation is mined, MRFP-Tree for secondary relation in Table 6 is created. Please note that the order in which the relations are processed is insignificant as the processing is done independent of other relations. In this example, the primary relation is processed first simply for explanation.

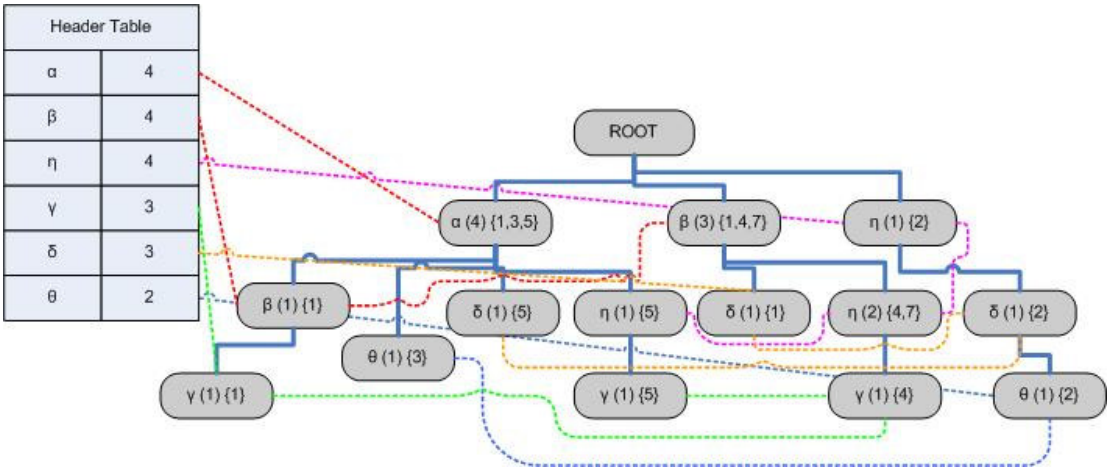

Figure 6: MRFP-Tree for Secondary Relation 1 (Table 6)

Figure 6 shows the MRFP-Tree for the secondary relation when support ≥ 2. Mining this tree yields the frequent patterns 9-12 shown in Table 9. The patterns 1-8 in Table 9 were obtained by mining the primary relation MRFP-Tree.

Table 9: Frequent Patterns table after mining MRFP-Tree in Figure 6

ID	ID set	Frequent Patterns	Support
1	2 6	H G	2
2	2 5	G E	2
3	1 4	D C	2
4	1 4	D C A	2
5	1 7	D B	2
6	1 4	D A	2
7	1 4	C A	2
8	1 3	B A	2
9	4 5	γ η	2
10	1 4	γ β	2
11	1 5	γ α	2
12	4 7	η β	2

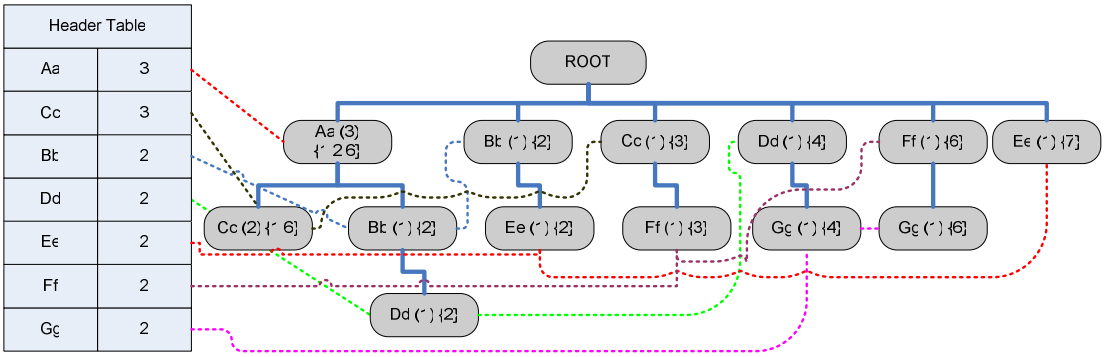

Figure 7: MRFP-Tree for Secondary Relation 2 (Table 7)

Figure 7 shows the MRFP-Tree constructed for Secondary Relation 2 with a support ≥ 2.

Table 10: Frequent Patterns table after mining MRFP-Tree in Figure 7

ID	ID set	Frequent Patterns	Support
1	2 6	H G	2
2	2 5	G E	2
3	1 4	D C	2
4	1 4	D C A	2
5	1 7	D B	2
6	1 4	D A	2
7	1 4	C A	2
8	1 3	B A	2
9	4 5	γ η	2
10	1 4	γ β	2
11	1 5	γ α	2
12	4 7	η β	2
13	1 6	Cc Aa	2

Once all the relations have been processed individually, they are joined through the frequent patterns obtained in Table 10. The ID set attribute in the frequent

patterns table is treated as an item field and the final MRFP-Tree made for this table is depicted in Figure 8.

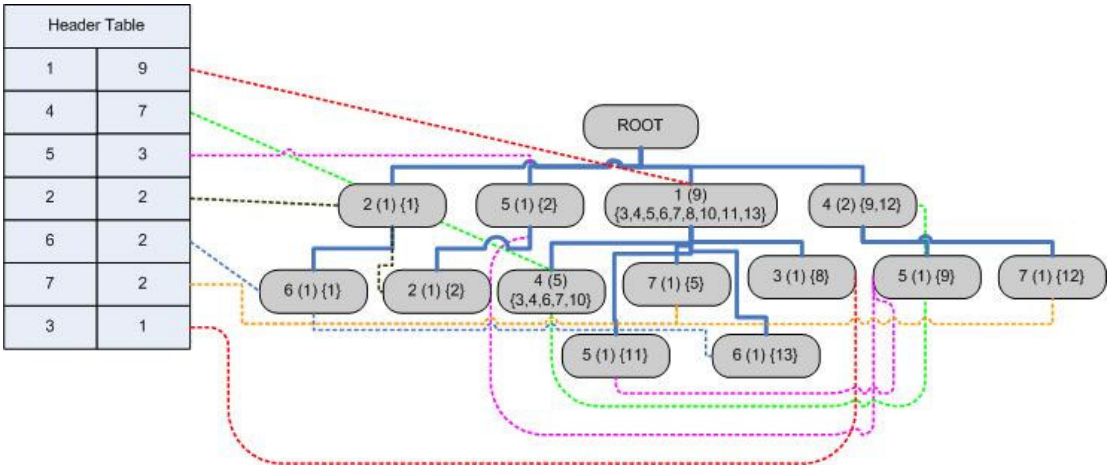

Figure 8: The Final MRFP-Tree

Mining this tree generates the frequent patterns that consist of IDs. These patterns are mapped on to real items using the ID_Item_Mapping procedure explained in the previous section. The ID column in Table 11 shows the frequent items obtained by mining the final MRFP-Tree for cross support ≥ 1. These IDs are mapped onto actual items to get the frequent patterns shown in the Frequent Patterns column in Table 11.

A rule generator program can be applied to the frequent patterns obtained above to get multi-relational rules.

Table 11: Final Frequent Patterns for the Sample Database

Cross Support	ID set	ID	Frequent Patterns
2	1 3	8	A, B
2	4 7	12	β, η
2	1 7	5	D, B
2	2 6	1	H, G
2	1 6	13	Cc, Aa
2	2 5	2	E, G
2	4 5	9	η, γ
2	1 5	11	α, γ
2	1 4	3 4 6 7 10	D, β, C, A, γ

4. Multi-relational Clustering

Clustering is the process of grouping objects into classes so that their intra-cluster similarity is maximized and inter-cluster similarity is minimized. The objects can either be binary, categorical or numerical type. The input to a clustering algorithm usually is a data matrix. A data matrix lists m attribute-values of each of the n object. Therefore the data matrix is an n x m matrix. The distance between any two objects is measured by finding the scalar difference between the lists of attribute-values of the two objects. So if i and j are two objects and there are m attributes in the data matrix, then the distance between two objects d_{ij} is given as follows:

$$d_{ij} = \sum_{k=1}^{m} D(i_k - j_k) \tag{4}$$

where, $D(i_k - j_k)$ is the distance between two k^{th} attributes of i and j objects, i_k and j_k. Given d_{ij}, a clustering algorithm makes clusters by minimizing an error function so that d_{ij} for i and j in same cluster is less and between i and j in different clusters is more.

In this chapter clustering in multi-relational domain has been explored. Multi-relational clustering problem can be reduced to high dimensional clustering problem in single relations as both suffer from the problem of having irrelevant attributes making clustering difficult. The problem of grouping items that may not have direct relationship between each other has also been visited.

4.1 Clustering High Dimensional Data

It is our belief that exploring the area of clustering high dimensional data may provide an insight into how the existing technologies can be remodeled to work in multi-relational settings. The idea is to investigate if techniques that handle large dimensions can be effectively "tweaked" to handle a variety of attributes in multiple relations. For simplicity, one can assume that a database containing a large number of attributes is a high dimensional database. It is extremely challenging to cluster high dimensional data because of two reasons [20]. First reason being that several attributes may be irrelevant or insignificant to effectively contribute to the clustering process and they reduce the tendency of getting good clusters on data. In classical machine learning this is the "feature selection" problem. The second reason, an extension of first, is that the distance to nearest neighbor becomes indistinguishable from the distance to the majority of points, as there are so many points in space.

Dealing with the first problem is more of a manual task. By specifying only the relevant attributes as input for clustering the first problem may be solved. There is a vast body of literature on feature selection that many researchers consider to be a useful exercise [54] [55]. However, if even after removing irrelevant attributes, a large number of attributes exist, the second problem arises. The second problem is therefore the main obstacle that almost all high dimensional clustering algorithms, dimensionality reduction algorithms and co-clustering algorithms attack.

Co-clustering is the problem of grouping both attributes and items simultaneously to generate clusters. This approach reduces the difficulty of clustering items based on attributes to clustering attributes based on items. Co-clustering is similar to clustering of categorical data. Categorical data contains values with no

inherent semantics. For example market basket data, as shown in Table 12 has this form. Every transaction in such a dataset is represented by enumerating all items j as attributes, and by associating with a transaction the binary attribute-value that indicates whether j^{th} item belong to a transaction or not.

Table 12: Categorical & High Dimensional Market Basket Data

Transactions	Pepsi	Chips	Cola	Bread	Milk
1	Yes	Yes	No	Yes	Yes
2	No	Yes	Yes	Yes	Yes

Such representation is sparse and two random transactions have very few items in common. This is why similarity between them is usually measured by the Jaccard coefficient [21] stated as follows:

$$J_{ij} = \frac{x}{x + y + z} \tag{5}$$

where,

x = number of variables that are positive for both the transaction

y = number of variables that are positive for i^{th} transaction and negative for the j^{th} transaction

z = number of variables that are negative for i^{th} transaction and positive for the j^{th} transaction

For the above example, x = 3, y = 1, z = 1, and J_{12} = 3/5. But as dimensionality increases, and number of common values becomes small, clustering methods based on similarity metrics are no longer effective.

In case of clustering transactional data, the problem is further aggravated. One of the several techniques that have been proposed to solve categorical clustering of

high dimensional data is based on a Hypergraph model [9]. Their approach is based on hypergraph model. In a hypergraph model each data item is represented as a vertex of a hypergraph and the related data items or attributes are connected with weighted hyperedges. They reduce the problem of clustering in high dimensional spaces to clustering related items. They first find association rules in the database, then create a hypergraph of these rules and perform a k-way partitioning of the hypergraph to obtain clusters of related items.

A similar approach was presented by Raghavan et al in their STIRR (Sieving Through Iterated Reinforcement) method [22] that uses spectral graph partitioning for categorical clustering. In STIRR a dynamic system instead of association rules formalize the co-occurrence. The main drawback of this approach is its convergence speed which can cause it to be extremely slow. There are several other clustering techniques available for high dimensional data [54][55][56].

Concluding this discussion, it is clear that clustering high dimensional data requires some sort of co-occurrence analysis of the items before the data is partitioned. For clustering multi-relational data too, the co-occurrence of items needs to be formalized.

4.2 Grouping related attributes to formalize multi-relational co-occurrence

Clustering algorithms quantify the extent of the similarity between two ‘observations’ or ‘data points’ or ‘tuples’. Often, distance metrics are derived, for example, in the Euclidean space to optimize the separation between data points. The clustering algorithm then attempts to reduce some error criteria using an objective

function such as entropy [23] [24] or the divergence between two distributions [25], etc.

Along similar lines, one can divide the task of grouping related attributes down into two sub-problems, (a) the task of first quantifying the notion of similarity and (b) the subsequent formation of groups, retaining attributes similar enough in the same group based on the derived metrics.

Table 13: Categorical Representation

Automobile	Color	Class
Toyota Camry	Red	4-Door
Mini-Cooper	Green	2-Door

Table 13 shows an example of a categorical dataset that describes two makes of cars. The following groupings of the attributes are immediately clear, {Red, Green}; {4-Door, 2-Door}; {v6, v8}. Now the question is how does on quantify the distance between the concepts Red and Green relative to 4-door? This is possible using external probes, which will be discussed shortly.

Using entropy as a clustering criterion, one tends to minimize the system entropy,

$$E(\overline{C}) = \sum_{k} \left(\frac{|C_k|}{|D|(E(C_k))} \right) \tag{6}$$

where the database D is partitioned into Ck clusters

provided,

$$C_i \subset D$$

and,

$$C_i \cap C_j = \phi, \forall k$$

Turning on to techniques that have been developed to explore the relationships between attributes in large databases, specifically techniques such as the external probes approach mentioned before. External probes [26] [27] is an external measure of the similarity between two attributes. The measure is determined with respect to a subset of the other features. The basic idea is that in a 0/1 relation r, the two attributes A and B are similar if their sub relations $\sigma_{A=1}(r)$ and $\sigma_{B=1}(r)$ are similar. This notion of similarity is dependent on a limited set of attributes called probes. External probes satisfy some of the basic requirements of a good distance metric [28].

- It is symmetric d(A,B) = d(B,A).
- The distance is 0 if and only if the attributes are identical, this requirement is subject to the application.
- It satisfies the triangle inequality, d(A,B) + d(B,C) $\geq$ d(A,C)
- It captures the notion of similarity. As the similarity between two attributes increases, the metric tends to zero.

Some of the basic problems that plague association rules are overcome in this case. Take for example the negative dependence between Coke and Pepsi in a set of transactional data. Since we examine all relations where both Pepsi and Coke occur in transactions, with respect to other attributes (for example chips, popcorn, pizza and other items of that nature), we will find clear patterns in transactions indicating that the two beverages are purchased under similar circumstances. For example, when shopping for party supplies, there is a high probability that the customer may buy

either beverage. This observation indicates a strong notion of association between the beverages. There are practical limitations when using this metric. For one it is only applicable on market-basket like and discrete datasets. In the past, agglomerative clustering methods have been used in conjunction with this method to successfully recover clusters [26]. There are also other examples of methods that perform grouping in Clustering, co-clustering, association rule-mining, feature selection and reduction techniques and Dynamical systems that require inter-entity distances.

4.3 RICH: multi-Relational Item Clustering using Hypergraphs

This thesis presents a new multi-relational item clustering approach based on hypergraphs called RICH. After detailed investigation in areas of clustering high-dimensional data and attribute grouping, RICH has been designed to cluster relational data so as to establish co-occurrence of items using frequent patterns generated from MRFP Growth and then partitioning the items using hypergraph partitioning algorithm HMETIS [29].

4.3.1 Hypergraphs

A hypergraph H = (V, E) is a generalization of a graph, such that each hyperedge E may be connected to more than two vertices V. Let $X = \{x_1, x_2, \ldots, x_n\}$ be a finite set, and let $\xi = (E_i \mid i \varepsilon I)$ be a family of subsets of X. The family ξ is said to be a hypergraph [14] on X if

1. $E_i \neq \Phi \ (i \varepsilon I)$
2. $\bigcup_{i \in I} E_i = X$.

The couple H = (X, ξ) is called a hypergraph. | X | = n is called the order of this hypergraph. The elements $x_1, x_2, \dots, x_n$ are called vertices and the sets $E_1, E_2, \dots, E_m$ are called edges.

An illustration of a hypergraph is shown in Figure 9. An edge E_i with $| E_i | > 2$ is drawn as a curve encircling all the vertices of E_i. An edge E_i with $| E_i | = 2$, is drawn as a curve connecting its two vertices. An edge E_i with $| E_i | = 1$, is drawn as a loop in a graph. Two vertices said to be adjacent if there is an edge E_i that contains both of these vertices. Two edges are said to be adjacent if their intersection is not empty.

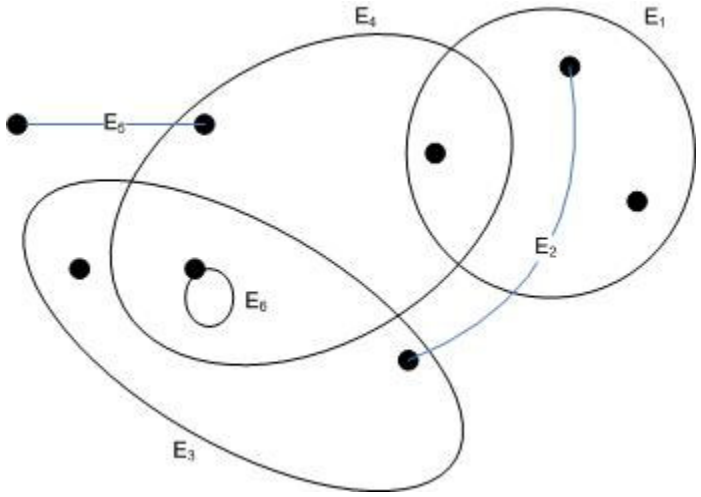

Figure 9: An illustration of a Hypergraph H

The above hypergraph (X: E1, E2, E3, E4, E5, E6) is a connected hypergraph of the order 8 with a cycle. If there is a chain in the hypergraph that starts at vertex a and terminates at vertex b, the relation a ≡ b is an equivalence class whose classes are called connected components of the hypergraph. If C1 is a connected component that intersects edge E, then C1 contains E.

4.3.2 RICH approach

Given a set of relations R_n that belong to a database D, first, frequent patterns across these relations are obtained. To generate frequent patterns the relations R_n, along with a set of attributes from each relation, support counts and cross support are given to the MRFP_Growth method. The multi-relational frequent patterns generated by MRFP_Growth are in the form of frequent itemsets and cross support values.

The cross support values indicate the frequency with which each itemset occurs in D. This statement means that the itemsets obtained from MRFP Growth are actually rudimentary forms of clusters of items that occur together frequently in the entire database. Therefore, these itemsets are the best representatives of the entire database.

The next step is to transform these patterns into a hierarchy. One way of achieving it is by transforming them into a weighted graph or tree structure. The motivation behind this step is that frequent patterns are usually in the form of combinations of items, such as $\{A, B\}_2$, $\{A, B, C\}_5$, $\{B, C\}_6$, $\{A, D\}_4$. For making clusters these patterns can be combined together with each item in pattern behaving as the vertex and each edge weighed by the cross support or confidence.

So now there are two questions that need to be clarified:

1. Should graphs or hypergraphs be used for representation of this information?
2. Should cross support or confidence be used for weighing edges?

Here is an explanation for the first one. An itemset may consist of two or more frequent items. Therefore a two dimensional representation of the itemsets and their

support values can be achieved by constructing a suffix tree (a form of graph with no cycle) of the items. For example if {A, B, C} (2), {A, D} (3) and {B, C} (2) is a set of frequent patterns, then the 2-D representation of this information is given as follows:

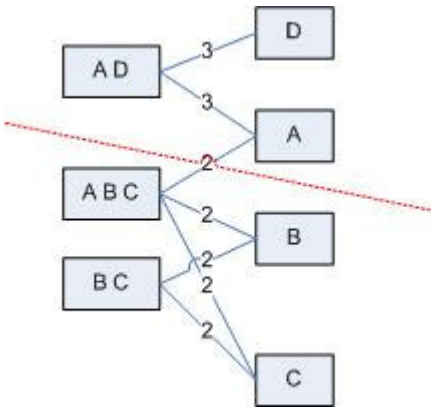

Figure 10: A Suffix Tree representation for the Frequent Patterns.

Each item of a frequent itemset is a leaf in the suffix tree. A frequent itemset is an internal node of the tree and each edge has the cross support as weight. The dotted red line indicates a partition of the tree using a partitioning algorithm. This 2-D representation has several drawbacks.

1. It unnecessarily requires more space for storage of patterns obtained by combining different items
2. It fails to show any relationship between {B C} and {A B C}.
3. After partitioning of the tree, the item {A} belongs to say partition 1 and the itemset {A B C} belongs to partition 2, therefore to which partition does the item A belong is still uncertain.

Therefore a hypergraph is a better representative of the frequent itemsets generated. In RICH model, each frequent itemset is a hyperedge E of a hypergraph H = (V, E). Then a vertex in H is each distinct item in the itemset.

Answering the second question, a hyperedge indicates the affinity of each item with the other items with respect to the database. The clause "with respect to the database" is important to be reviewed carefully. The clause takes into consideration the global frequency of occurrence of the pattern. The global frequency is needed, as patterns from different relations may be combined together. Global frequency is given by the *cross support* metric. [9] uses *confidence* for weighing the edges. *Confidence* is the conditional probability of occurrence of one item given another item in an itemset. It does not measure the global outlook of an itemset. Therefore RICH uses *cross support* for weighing edges as opposed to using *confidence*.

Once the hypergraph is constructed, it needs to be partitioned to give clusters of different items. The problem of hypergraph partitioning is to use a *min-cut hypergraph algorithm* that partitions the vertices of a hypergraph into n parts, such that the number of hyperedges (or weight of hyperedges) connecting vertices in different parts is minimized. The formation of a cluster needs minimized inter-cluster item connectivity and maximized intra-cluster item connectivity. Thus a hypergraph partitioning algorithm based on min-cut strategy, HMETIS [29] is used for partitioning items to generate final clusters.

Following is an example given showing RICH's approach. The example given in the previous chapter for MRFP_Growth is continued here. A hypergraph is constructed on the frequent patterns table given below:

Table 14: Frequent Patterns & Cross Supports

Cross Support	Frequent Patterns
2	A, B
2	β, η
2	D, B
2	H, G
2	Cc, Aa
2	E, G
2	η, γ
2	α, γ
2	D, β, C, A, γ

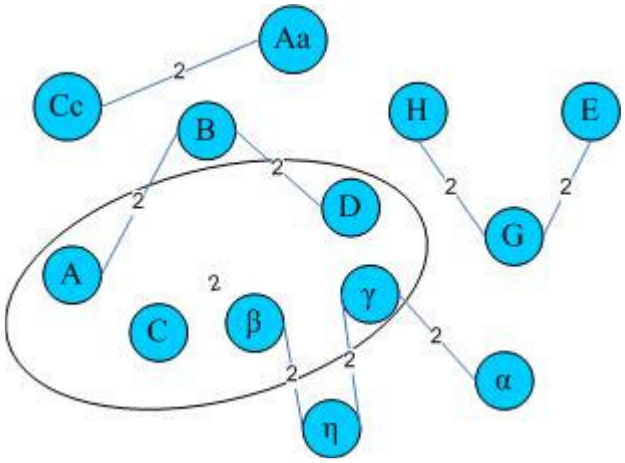

Figure 11: A Hypergraph for the frequent itemsets in Table 14

Figure 11 shows the hypergraph on the itemsets. It has 9 edges and 13 vertices. Each hyperedge is labeled using the cross support. HMETIS is a multi-level hypergraph partitioning algorithm. It uses the min-cut strategy to minimize the weighted hyperedge cut to create partitions with high vertex connectivity in each partition, thus resulting in good clusters.

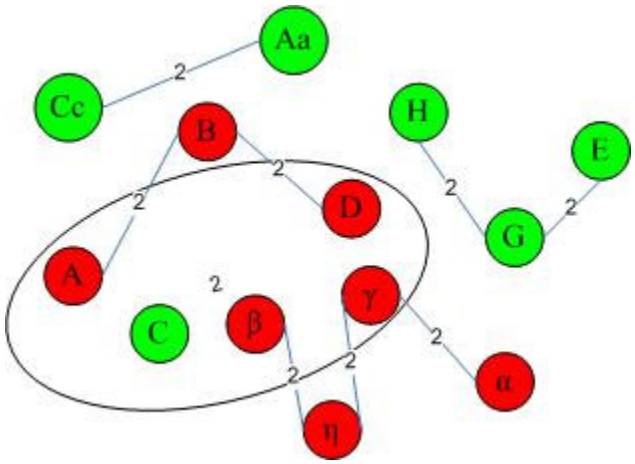

Figure 12: Hypergraph partitioned into 2 using HMETIS

Figure 12 shows the clusters made by HMETIS on the hypergraph shown in Figure 12. The items [A, B, D, η, γ, β, α] are partitioned into a single cluster while the items [Cc, Aa, H, G, E, C] are placed in another cluster. Thus this is the way multi-relational clustering is achieved.

Extensive empirical evaluation of RICH was performed and the results are given in the Experiments and Results chapter. The results show that RICH makes good quality clusters on both single-relational and multi-relational data. RICH was compared to CLUTO that is also a hypergraph based clustering algorithm.

5. COMMA: Combining Multi-relational Multimedia for Associations

5.1 Introduction

It is said that a picture is worth a thousand words; but determining the likely words that constitute the correct description of the picture is considered to be a challenging problem by the computer vision, text mining, and the multimedia data mining communities. Knowledge derived from these two domains i.e., image and text data together is more descriptive compared to when each domain is considered in isolation from one another. Based on this fact, it is our conjecture that multi-relational associations should capture more information from the combined metadata. Conventional approaches use metadata from individual image features or text domain annotations using a relational join and develop feature based clusters. This work describes a method of formulating this conjecture as a multi-relational hypothesis and tests the validity of integrated mining of combined multimedia data using multi relational association rules.

Recent years have witnessed a phenomenal growth in image databases and retrieval systems such as Viper [30] and MultiMediaMiner [31] to name a few. The World Wide Web has emerged as the largest repository of image data in the world. Image retrieval based on keyword search from such large databases poses a significant challenge. The search results can be greatly improved if the images are already annotated. However, owing to the large number of images in these databases,

the only viable means to annotate images is to automate this process, since manual annotation can be a tedious and expensive job.

The problem of auto annotation is usually treated as a supervised learning problem where higher-level features are extracted from images and complex object detection algorithms are employed to generate keywords. Image segmentation and labeling of objects is not easy. Several clustering and classification techniques have been employed for auto-annotation of images [32] [33] [34] [35]. The "blob" approach requires the images to be in a state where object recognition is possible [33] [35]. Hsu et al employed the idea of viewpoints, which refer to the notion of invariant relationships between objects in an image [36]. Object identification in images is usually expensive and thus increases the cost of auto-annotation [33] [37] [38].

Association rule mining for images is a fairly nascent subfield of image mining. There are two main approaches for association rule mining in images. The first one involves just mining images while the second one involves mining images along with some textual data associated with the images. We apply the latter approach in this paper. We extract basic features like color, orientation and intensity from the images. Features that are more complex were specifically not preferred so as to study and analyze the performance of the multi-relational approach with a minimum consideration for semantics of the image. The low-level features considered are color (number of pixels that are red, green, blue and yellow), orientation (edge orientations of degree 0, degree 45, degree 90 and degree 135) and intensity. The image features are extracted based on the focus of attention theory initially proposed by Itti and Koch [39]. The selective attention model allows the system to concentrate on processing

salient objects in the scene without the need to process the unimportant aspects. The attention model processes the input image in three parallel feature channels: intensity contrast, color and orientation channels. The feature saliency maps topographically represent the saliency of objects in the scene based on respective features. For a detailed description of still-feature extraction from images, refer to our previous work [40].

We restrict the images under consideration to certain categories, such as "flowers" and "lakes and mountains" or their combinations. This allows us to exploit the correlation between low-level features in an image and high-level semantic content without object identification in the image. It is our hypothesis that within categories of images, enough similarities exist to allow the discovery of multi-relational associations, which can be used for auto annotation of images. Another motivation for using low-level features is the need to maximize system throughput while minimizing the overhead cost of storing high-level features. However, the application can be scaled to include high-level features such as shapes and objects by designing appropriate tables to hold them. Notice that the application of this framework to specific domains implies that it should not be used for an open-ended domain such as "nature" (which may comprise of landscapes, flora and fauna, fruits and vegetables, underwater images etc.)

Consider the scenario where images are stored in a database and associated with these images are annotations or captions that are derived from multiple sources. Although the annotations can be stored within the same database in different tables or combined together into a single table, the upshot of the latter approach is that it does

not take into account that there can be several different annotations for the same image depending upon the users (or the intelligence of the auto-annotating system annotating the image.) Hence, there is a one-to-many relationship from the image domain to the text domain. Doing a simple join can be expensive if M is a big number in 1: M relation between the tables and such a join would be unnecessary if the tuples in a table do not qualify as frequent patterns. The relation between annotations in multiple tables and image features can however, be captured by multi-relational association rules thereby getting formulated as a multi-relational mining problem. Since, this is the motivation behind our project, we termed the framework CoMMA: Combined Multi-relational Multimedia mining using Associations.

5.2 Image Annotation & Retrieval

Image annotation can greatly enhance image retrieval. Many annotation schemes have been proposed for faster and better image retrieval. The subfield of Content Based Image Retrieval (CBIR) employs global features of images such as color histograms and was used in IBM's QBIC (Query by Image Content) [41] and also in region based approaches involving "blobs" [33] [35]. FAST (Fast and Semantics-Tailored Image Retrieval Methodology) [42] uses fuzzy logic to create a new indexing method HEAR (Hierarchical Elimination-based A* Retrieval) to handle the region based image information consisting of colors, texture and shape. Relevance Feedback (RF) analysis is another effective solution for CBIR. Semi-Automatic Image Annotation [43] depends on user feedback and adds successful image search keywords as annotations to images. Zhong et al. introduce PCA [44] to reduce the noise in original images and the dimensionality of the feature spaces. Authors of

MultiMediaMiner [31] mention about the usage of association rule mining to get rules based on colors of CT scan as an interesting application. Monay et al. compare [45] simple Latent Space Models for the task of annotations.

Image annotation has been mostly studied from a statistical as well as supervised learning perspective [46]. The multi-relational association rule approach is somewhat similar to the statistical approach in that the support and confidence of rules across the image and text domain are related to the statistical distribution of features, while relations capture more information than just simple statistical links. Two related subtasks are involved in recognition of images i.e., auto annotation of images that involves recognizing whole images and object recognition, which involves recognizing objects in the images. In this paper, we address the former task using minimal features from the image domain. We would also like to note that the problem that we are addressing is not only that of finding a particular suitable annotation for an image but also that of finding a set of keywords that can be used as query-set by a human for retrieval using a search engine.

5.3 CoMMA Feature Extraction

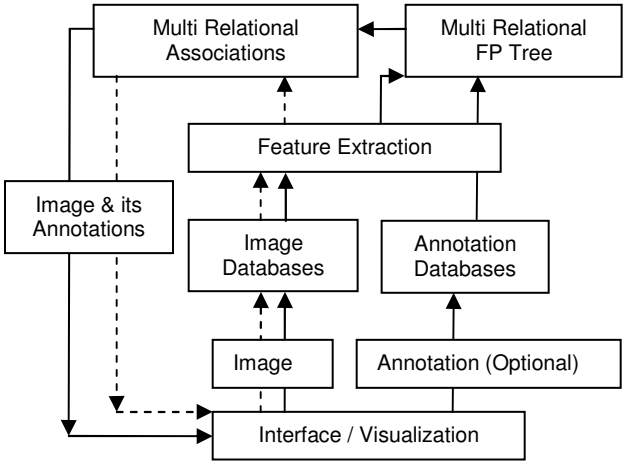

Figure 13: The CoMMA Framework

CoMMA is developed as a general framework for employing multi-relational association rules for auto-annotation of images in specialized domains. In section 4, we describe a multi- relational version of the FP-Growth algorithm, which forms the core of the current application. In CoMMA, a user may optionally upload an annotated image. Such an image is used for generating possibly new rules indicating that the system has learnt something new (training mode). Otherwise, rules are not generated but are used to annotate the given image (test mode). Figure 13 gives an overview of how CoMMA works. Starting from the User Interface, images are uploaded into the image database along with the corresponding annotations. Low-level features are extracted from the image while the text features mainly consist of terms from the annotation or caption. Multi-relational association rules are generated by applying the MRFP-Growth algorithm. Finally, these rules are used to annotate test images and the performance is compared with the original ground truth. When an annotation is not provided for an image, the image features are extracted and annotations are obtained from the previously generated association rules as shown by the dotted line in Figure 13.

As described in [40] and mentioned previously, the feature extraction is based on Itti and Koch's focus of attention theory [39]. Given a scene, humans selectively attend to important salient regions of the scene. For example while driving on a road, the red and yellow street signs stand out in the scene. The focus of attention algorithm used in this system, processes an image to extract the color, orientation and intensity features. The four color saliency maps (Red, Green, Blue, and Yellow), orientation (0 degree, 45 degree, 90 degree, 135 degree) and intensity contrast maps are feature

saliency maps. Gaborski et al. experimented [40] with feature saliency to infer the importance of each feature based on different scene type. Their approach collected human eye-tracks for images of natural landscape scenes, indoor scene, building/city scenes and fractal images. The eye-tracks collected for the four scene types were used to find the feature that dominated the subject's attention. Based on the correlation studies on multiple images, intensity contrast gave the highest correlation for natural scenes and building/city scenes. Color gave the highest correlation for indoor and fractal scenes. Based on these observations a test image can be classified as being one of the four scene types. These studies demonstrate the efficiency of low-level features in classifying scenes and motivated us to use low-level features for generating rules for image annotations.

We treat each image i_k in the database as a pseudo-vector that consists of the nine features described in section 1. The vector space consists of all image features. The image features were further discretized, so that the final image feature vocabulary consisted of more than 2700 feature terms. In the image mining domain, the modeling of image annotations has usually been done by the concatenation of image feature vectors and a feature vector of words [36]. If there are one-to-many relationships between the image feature vector and the term vectors, the same technique (concatenation) can still be used but the cases where terms in a description are further related to other terms, as is the case in the current problem, cannot be handled without loss of information. This was the main motivation for keeping the images and the annotations/captions/descriptions in separate tables.

If the task is just coming up with a set of words to be used by a human expert for annotating images, then clustering can also work quite well.. However, the problem with employing clustering for this task is that individual clusters usually have a much larger data spread as compared to association rules which are comparatively straight forward. Clustering was thus not employed in this application and is being explored for a baseline comparison with our approach.

5.4 CoMMA Results & Discussion

5.4.1 Data Organization

Although research in auto-annotation of images has been going on for several decades, standardized datasets have not come into existence. It consists of images from different sources like Corel Professional Photo CDs; University of California Berkeley Floral images[2]; University of Washington ground truth dataset[3], snapshot[4], Freefoto[5], United States Fish and Wildlife Services National Image Library[6]. Multiple sources were used instead of concentrating on a singular source to ensure that the results are not already biased because of the dataset. These summed up to 2036 images. Numerous human experts were asked to annotate the images to the effect that there was some overlapping between the annotations given by these experts.

[2] http://elib.cs.berkeley.edu/photos/tarlist.txt
[3] http://www.cs.washington.edu/research/imagedatabase/groundtruth/_tars.for.download
[4] http://www.snap-shot.com/pages/land/
[5] http://www.freefoto.com
[6] http://images.fws.gov

Figure 14 gives the empirical distribution of keywords for the top 65 keywords with the highest frequency. For this particular dataset, 20 keywords accounted for about a third of the probability density mass function.

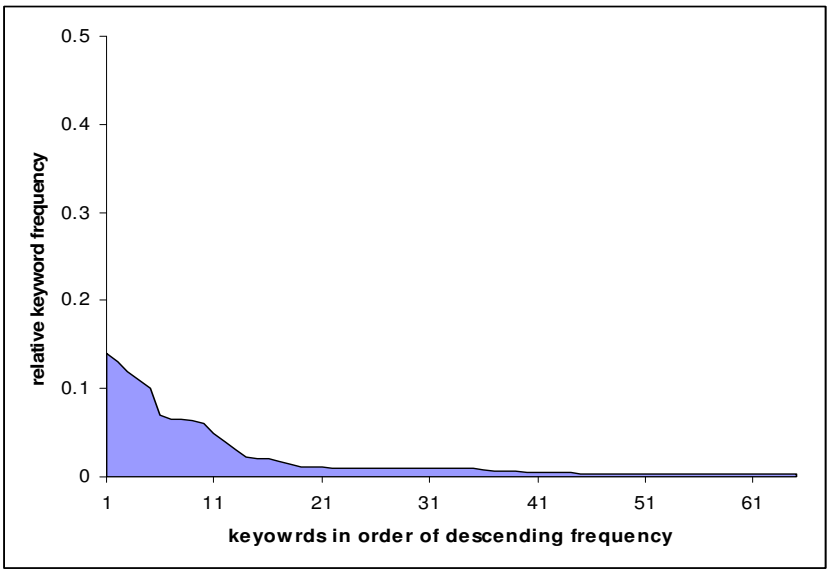

Figure 14: Empirical keyword distribution in sample data set.

The relations (tables) that were used are given in Table 15, Table 16 and Table 17. Image Table (Table 15) is the primary table (See Definition 4.2.1) with image-id (primary key) and absolute path to the location of each image on the disk. Table 16, Annotation_English, is a foreign table (See Definition 4.2.1) and has attributes image-id (foreign key) and annotation. The table was named so, to allow creation of tables for annotations in other languages, which would make CoMMA multilingual. Finally, Feature table (Table 17) is another foreign table that holds image features and image-ids (foreign key).

Table 15: Image table (Primary table)

Image-id	Image Path
1	C:\34.jpg
2	C:\people.jpg
3	C:\sky.jpg
…	…

Table 16: Annotation_English table (Foreign table)

Image-id	Annotation
1	flowers leaves
1	Sky flowers
2	People sky flowers
3	Sky clouds flowers
3	People flowers clouds
…	….

Table 17: Feature table (Foreign table)

Image-id	Features
1	258R 92G 44B 57Y
2	768R 92G 33B 18Y
3	457R 92G 77B 57Y
…	…

The features have been normalized to account for different image sizes. Each image feature has been appended with a distinguishing tag, which helps in generating rules.

Table 18 gives a list of the tags that were used and their meaning.

Table 18: Tag Listing

Tag	Stands for
R	Red
G	Green
B	Blue
Y	Yellow
D0	Edge Degree 0
D45	Edge Degree 45
D90	Edge Degree 90
D135	Edge Degree 135
I	Intensity

Some of the rules that are generated on running MR FP-Growth are given below:

48D45 → WEEDS

BOAT → 36R, 108Y, 22D135

48D45 → BUSH, DARK

48D45 → GRASS

4Y → EARTH, GROUND

4B → POPPY

SKY → 100B, 108D45

SEA → 218B, 55D90

4B → DARK, NIGHT

339G → DIRT

339G → SHRUBS, GRASS, LEAVES

370R → FLOWERS

370R → VINES

29D0 → MOUNDS

Using such rules, new images are auto-annotated and the results are discussed in the next section.

5.4.2 CoMMA Experiments & Results

Table 19 gives some of the results obtained on auto-annotating test images. To evaluate our results we used the metric given below.

Table 19: Sample results describing images and their corresponding annotations

	**Original Keywords:** BUSH SMALL FLOWERS GROUND SMALL **CoMMA Generated Annotations:** RADIANT BUSH SMALL GRASS LEAVES FLOWERS BERRIES GROUND WEEDS EARTH DRY DESERT
	**Original Keywords:** BUSH FLOWERS GROUND **CoMMA Generated Annotations:** BUSH SMALL OVER FOREST RIVER GROUND TREE LEAVES BUSHES GRASS SKY FLOWERS TREES ROCKS
	**Original Keywords:** LAKE SUMMIT IN ALASKA **CoMMA Generated Annotations:** STILL LAKE SUMMIT ALASKA NEAR ROCK ROCKS DESERT MELTING WALLS ICE

No common universally adopted benchmark evaluation metric exists for image annotation. Hence, different people assess the quality of annotations differently. However, the vocabulary statistics must be taken into account while evaluating performance since a poor system can just 'guess' the correct answer in at least some of the instances. To distinguish between systems that simply use empirical word distribution of the keywords in the training set and systems that employ a more

systematic approach we modified normalized score measure provided by Barnard et al. [32].

It is defined as:

$$E_{NS} = \frac{\left(\sum_{i=1}^{k} (r_i / n_i) \; [w / (N - n_i)] \right)}{k} \tag{7}$$

where n is the actual number of keywords in the test image, r is the number of correctly predicted keywords, w is the number of incorrectly predicted words, N is the vocabulary size and k is the total number of tables. For a system that predicts all the keywords correctly the value of ENS is positive one (+1.0), while for a system that predicts all keywords incorrectly the value of ENS will be a negative one (-1.0) and a for a system that just predicts all the keywords in the dataset for any image the value will be zero. The performance of our system is given in figure 6, against a pool of randomly selected images. The results are negative only in a few cases while it ranks higher than 0.5 for more than half of the data set.

In many specialized domains certain groups of keywords are more frequent than others. A sufficiently high frequency of such a group can significantly skew the results, making a quantitative evaluation of the results rather problematic. In order to make sure that this is not the case in the dataset that we are using, we gave the top 10 words as annotations to the test dataset. The performance was evaluated using the same metric; the results of these rounds of tests are given in Figure 15.

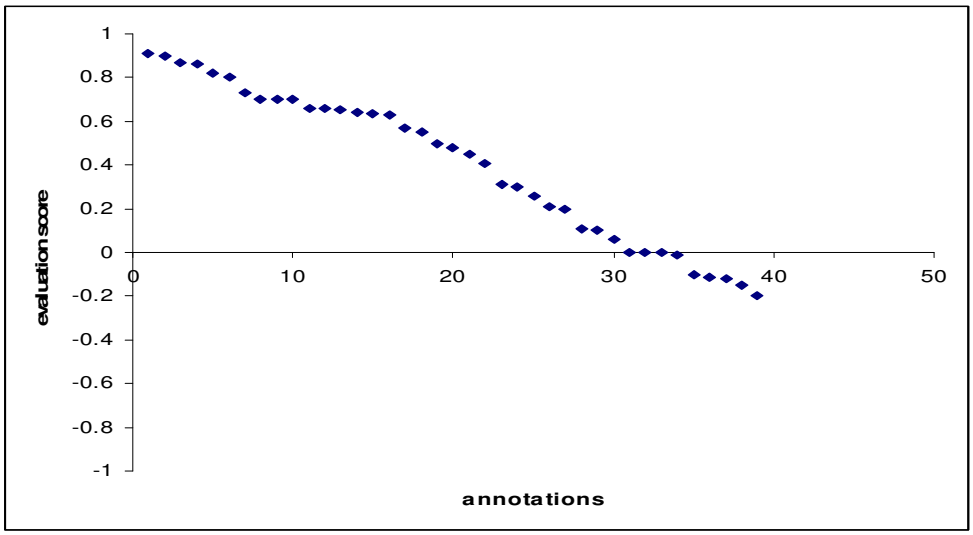

Figure 15: Evaluation score for the test dataset from highest to lowest for the maximum coverage of the dataset

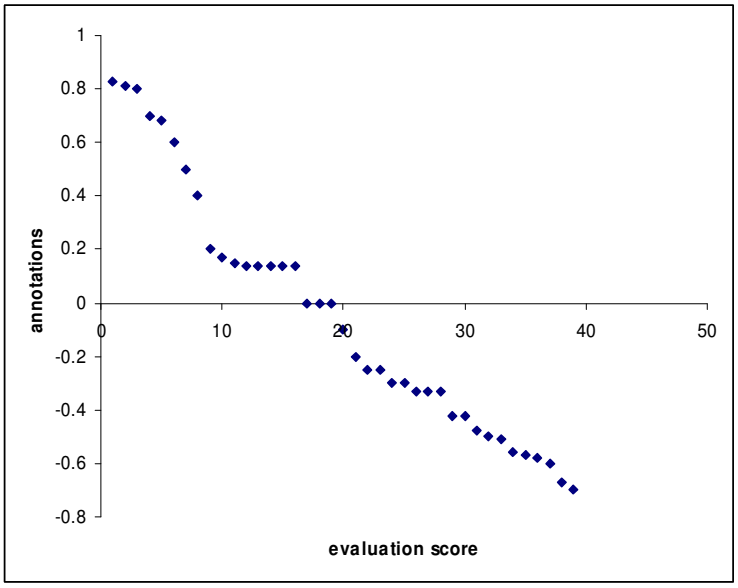

Figure 16: Evaluation score for the test dataset from highest to lowest for the maximum coverage of the dataset

Figure 16 clearly shows that annotating all the images with the top 10 frequently occurring words does not affect the result in any appreciable way and such a system ranks rather low performance wise. Random annotations seem to perform well for a

few images. The performance for most other images is fairly poor as is evident in Figure 15 as compared to Figure 14.

For more experiments that were conducted on CoMMA the reader is encouraged to go through the experiments section in [47] [48].

6. Experiments and Results

This chapter details the experiments conducted and results obtained to show the validity of MRFP Growth and RICH as multi-relational itemset mining and clustering algorithms respectively. The evaluation metrics, used for validation of clustering results, have been discussed at later in this chapter. Experiments were setup to verify the following claims made in the previous chapters:

- MRFP Growth algorithm finds association rules in a multi-relational scenario. To test this claim, MRFP Growth is compared against current state-of-the-art multi-relational association rule algorithms, WARMR [7] and Decentralized Apriori [8]. WARMR was downloaded as a part of the ACE data mining package[7] and a locally implemented version of Decentralized Apriori was used for the purpose of comparison.
- RICH is a multi-relational clustering algorithm, which finds groups of related items in a given database. Cluster quality of RICH was compared with that of the graph-partitioning based clustering algorithm CLUTO[8] [11].

The experiments were conducted on numerous datasets, both single relation and multi-relational. The experiments were run on a Dell machine with Windows XP Professional operating system, using Pentium IV running at 2.8GHz and 512 megabyte main memory. All datasets were stored in MS Access 2003. MRFP Growth has been designed to work with both Oracle and MS Access. The following

[7] http://www.cs.kuleuven.ac.be/~dtai/ACE/
[8] http://www-users.cs.umn.edu/~karypis/cluto/index.html

subsections inform the reader about the datasets used and the experiments that were conducted.

6.1 Datasets

Five datasets were used for conducting experiments, three of which are multi-relational and two are single table. Multi-relational datasets were used to verify that the algorithms MRFP Growth and RICH are able to mine interesting patterns and cluster them respectively out of a multi-relational database. The single table datasets were used for benchmarking performance of RICH.

6.1.1 PKDD Dataset[9]

The first dataset is a real-world multi-relational financial database used in PKDD CUP 1999. In this thesis a part of this database is used, which is relevant to the multi-relational mining that can be achieved by MRFP Growth. Its schema is shown in Figure 17. After schema modifications, the database consists of four relations. The relation, Account, being the primary relation has 4500 tuples. The Loan relation refers to Account, and has a class attribute, which labels the accounts as having good credit status or bad credit status. From this relation some positive tuples were removed to make the numbers of positive tuples and negative tuples more balanced. The Loan relation contains 324 positive tuples and 76 negative ones. The Trans relation was shrunk down to 1500 tuples.

[9] http://lisp.vse.cz/pkdd99/Challenge/

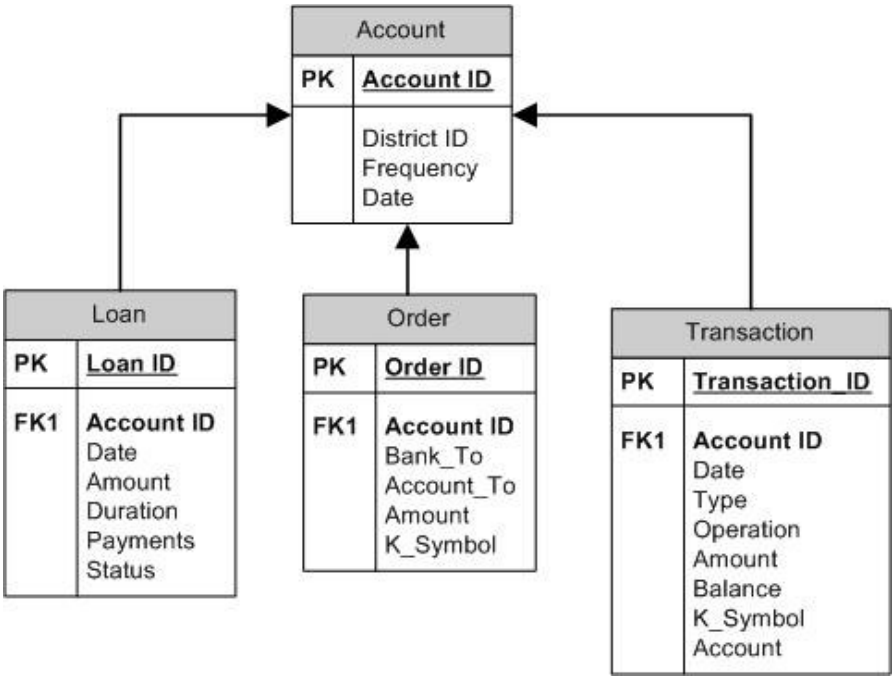

Figure 17: PKDD '99 Financial Dataset Schema

6.1.2 CoMMA Dataset

This is a multimedia database and was employed in CoMMA [47] [48]. This dataset and its composition are discussed in detail in the previous chapter. The database schema is shown in Figure 18. This database has three relations, Images being primary, Annotation_English and Image_features being secondary relations. Different human experts were asked to annotate all the 1532 images.

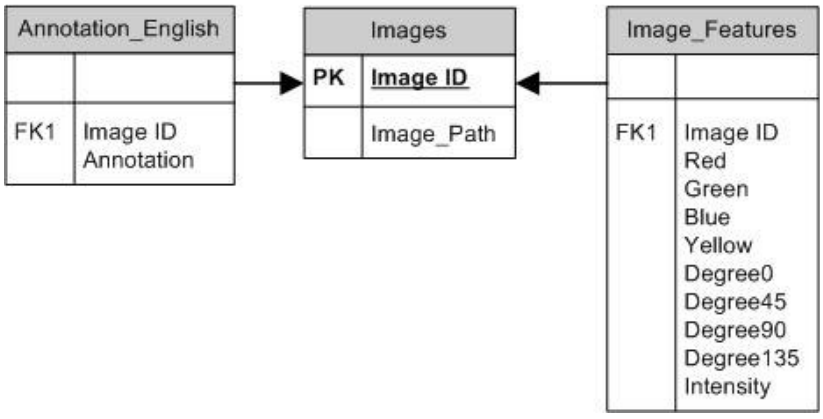

Figure 18: CoMMA Dataset Schema

6.1.3 Congressional Votes Dataset

This single relation dataset was obtained from the UCI Machine Learning Repository [50] and contains the United States Congressional Voting Records for the year 1984. Each record contains a Congressman's votes on 16 issues. All the attributes are boolean ("yes" or "no"), with a few of the votes containing missing values. The missing values are treated as another domain value for the attribute. Each record is labeled as "Democrat," or "Republican". There are 435 records in the set (267 Democrats and 168 Republicans).

6.1.4 Mushroom Dataset

The mushroom data set was also obtained from the UCI Repository [uci]. Each record describes the physical characteristics (e.g., odor, shape) of a single mushroom. There is a "poisonous" or "edible" field for each mushroom. All of the attributes are categorical and the set contains 8,124 records in all (4,208 edible mushrooms and 3,916 poisonous ones).

6.2 MRFP Growth Experiments & Results

This section deals with experiments and results of the MRFP Growth algorithm.

6.2.1 Performance Results

MRFP Growth as stated earlier finds frequent patterns from multiple relations. To achieve this, it needs a set of support counts for individual relations to be considered for pattern generation and the cross support value. The number of frequent

patterns generated, therefore depends upon this entire set of support counts and the cross support.

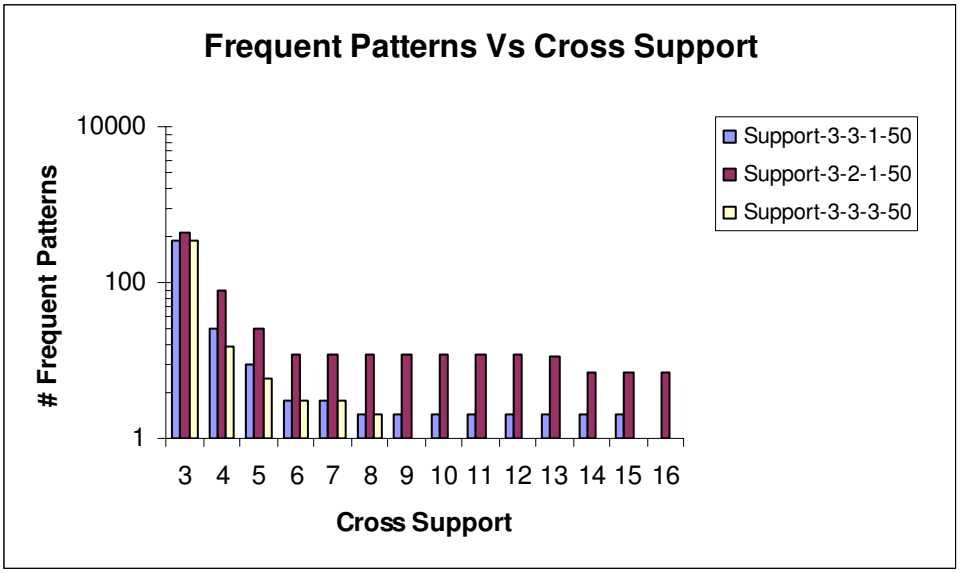

Figure 19: Frequent Patterns Vs Cross Support for PKDD dataset

Figure 19 shows a comparison of the number of frequent patterns generated for different combinations support counts on the PKDD dataset. Each bar series is named after the "support of accounts table-support of order table-support of loan table-support of transactions table". For example the series support-3-2-1-50 shows the number of frequent patterns generated when the cross support is altered between the range 3-16 and the support count for account table is 3, for order table is 2, for loan table is 1 and for transactions table is 50. The best support combination for maximum itemset generation on PKDD dataset is "Support-3-2-1-50".

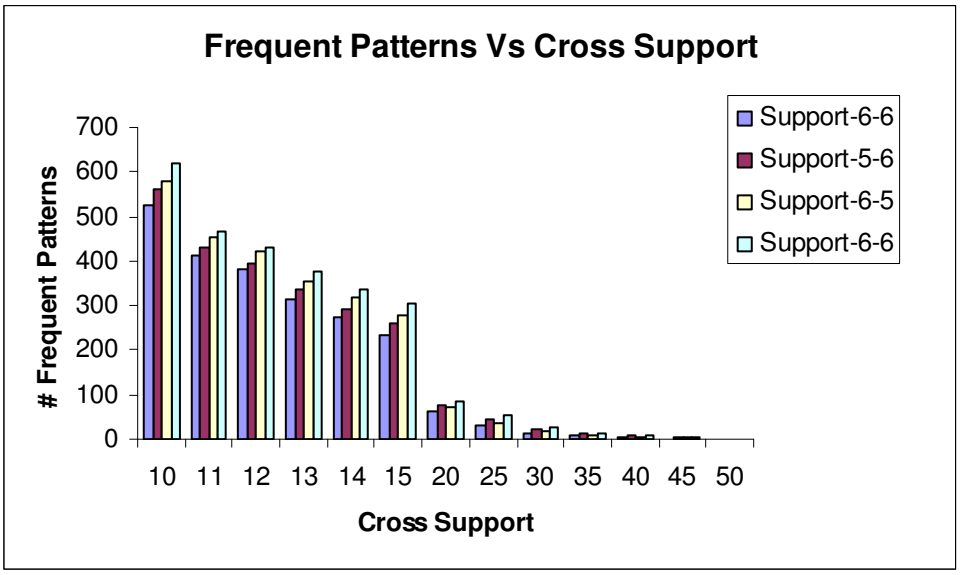

Figure 20: Number of Frequent Patterns Vs Cross Support for CoMMA dataset

Figure 20 shows number of frequent patterns generated Vs support values for the CoMMA dataset. In this graph "Support-6-5" indicates that the CoMMA dataset was mined for Support = 6 for the annotation table and Support = 5 for the images table, while the cross support was varied. Both the graphs indicate that as the support count is decreased, more frequent patterns are generated as more items qualify as frequent. And as the support is increased, the number of frequent patterns generated drops.

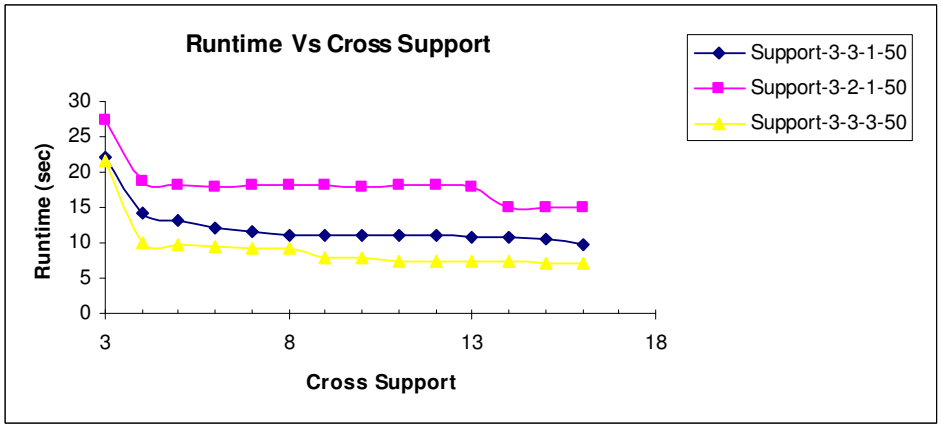

Figure 21: Runtime Vs Cross Support for PKDD dataset

Runtime performance for MRFP Growth on the PKDD dataset and CoMMA datasets is shown in Figure 21 and Figure 22 respectively. The series are named according to the different support count combinations used as explained earlier. The graphs show that as support increases, runtime decreases. The series with minimum support combination, Support-3-2-1-50 and Support-5-5 take maximum time to generate frequent itemsets for PKDD dataset and CoMMA dataset respectively.

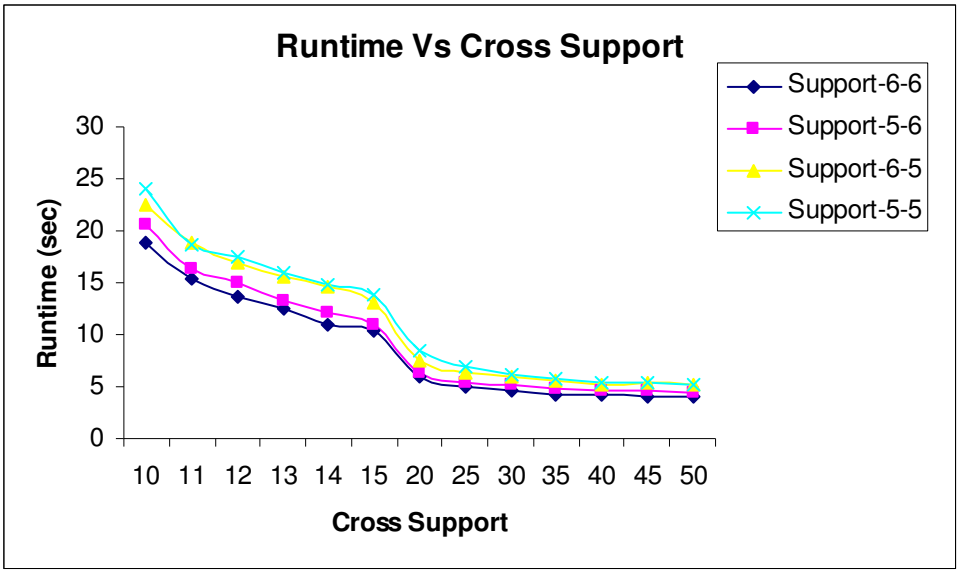

Figure 22: Runtime Vs Cross Support for CoMMA dataset

6.2.2 Comparative Evaluation

This section compares MRFP Growth to WARMR and Decentralized Apriori. Figure 23 shows the number of frequent patterns obtained by running MRFP Growth, WARMR and Decentralized Apriori on the PKDD dataset. Please note that the support measure as used in the graphs is cross-support for MRFP Growth.

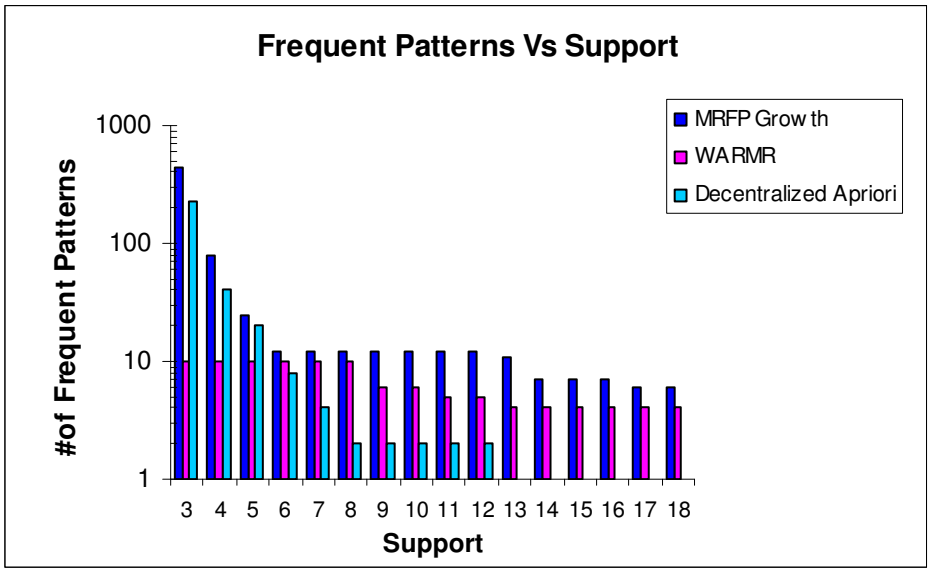

Figure 23: Number of Frequent Patterns Vs Support for PKDD dataset

It can be noted that MRFP Growth mines more patterns as compared to WARMR and Decentralized Apriori. The reason being that MRFP Growth considers patterns found in individual relations as frequent if they satisfy the cross support value, while WARMR builds up on frequent patterns found in target relation. The decentralized approach is almost the same as MRFP Growth's approach but is unable to mine as many patterns as MRFP Growth.

Figure 24 shows runtime comparison of the three algorithms. Runtimes of WARMR and Decentralized Apriori are higher as compared to the average runtime of MRFP Growth. Decentralized Apriori has a higher runtime initially as it generates a large number of candidate itemsets, out of which very few are actually frequent.

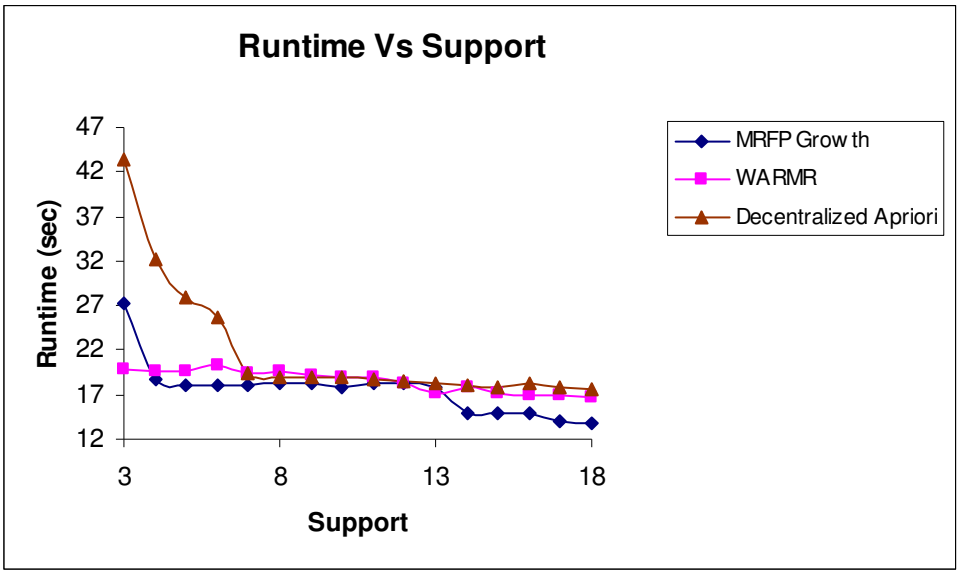

Figure 24: Runtime Vs Support

The next dataset on which comparison is based is the CoMMA dataset. WARMR has not been compared using CoMMA dataset as the annotations given in this set need to be converted into market basket data format before they can be applied to WARMR. Moreover, even if the conversion is done, the dimensionality of the data will be increased to such a huge extent (as the number of distinct keywords in annotations table is high) that WARMR will not be able to handle it.

Figure 25 and Figure 26 show the comparison of MRFP Growth with Decentralized Apriori for the CoMMA dataset. As expected, the runtime and number of frequent patterns generated for Decentralized Apriori is more than that of MRFP Growth for different support counts. For the earlier experiment and this experiment, the best support combinations were used for MRFP Growth.

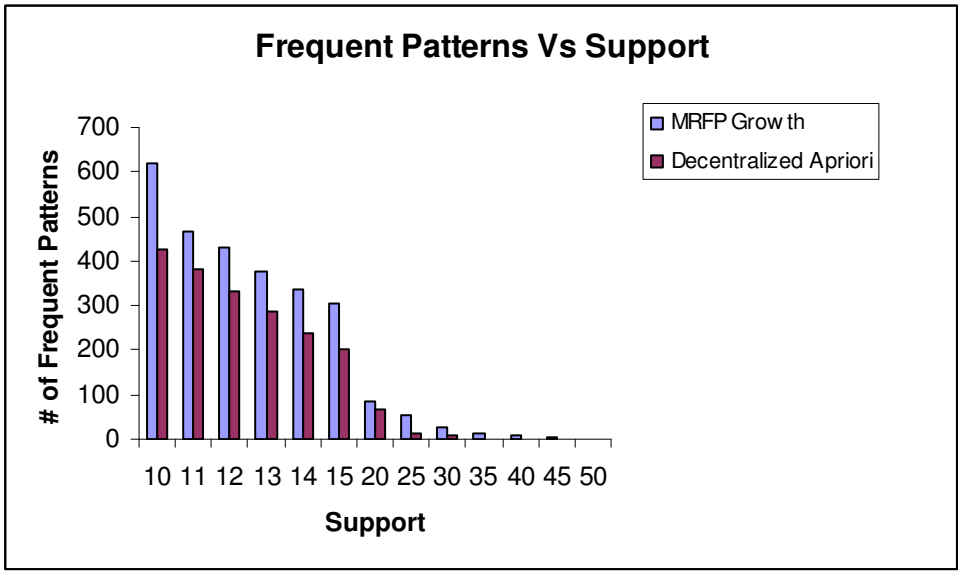

Figure 25: Frequent Patterns Vs Support for CoMMA dataset

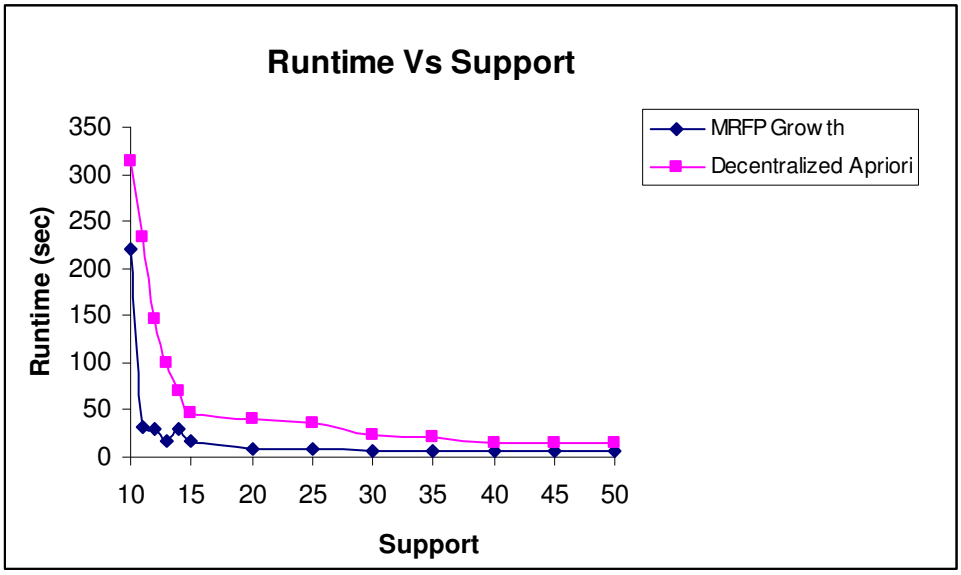

Figure 26: Runtime Vs Support for CoMMA dataset

6.2.3 Summary of MRFP Growth Results

These results show that MRFP Growth scores over the two techniques in the number of frequent patterns generated and runtime. Optimizations like cross support consideration at the time of pattern generation from individual tables and automatic

selection of support counts for individual tables, may further improve MRFP Growth's performance.

Current limitation of MRFP Growth is that it can only mine frequent patterns across primary and secondary relations. Relations of the form secondary relation B referring to a primary relation A, and another secondary relation C referring to B cannot be mined by MRFP Growth. While, in such cases relations C and A have weak semantic links, none the less it would be interesting to mine such relations and see what kind of patterns one may get or may not get at all. Figure 27 shows the possible joins in a database. The green dotted lines indicate joins that MRFP Growth can perform. The red dotted lines indicate the joins that MRFP Growth cannot handle and are weak semantic links.

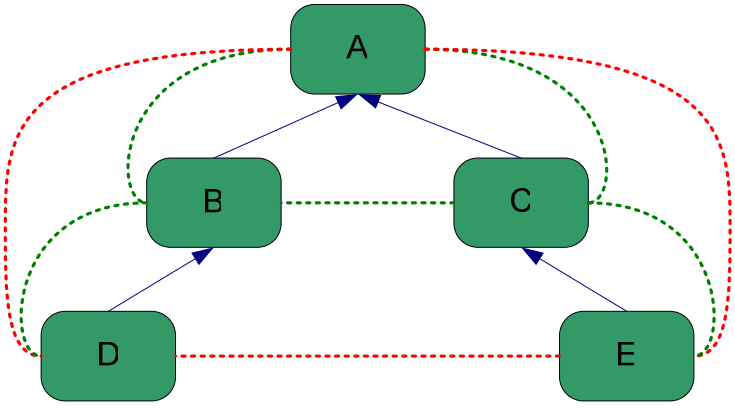

Figure 27: Possible Joins in a Database

6.3 RICH Experiments & Results

This section is dedicated to all the experiments and results that were run for evaluating RICH's performance and quality of clusters generated by it.

6.3.1 Cluster accuracy evaluation metric

This sub-section illustrates the difficulties encountered in evaluating the quality of the clusters generated by RICH. Different clustering algorithms use different evaluation metrics. [9] uses the same approach for item clustering but does not use any metric for evaluating quality of clusters, instead they have their results analyzed by human experts in the pertinent field. Many traditional clustering algorithms use Categorization Utility [51] and entropy based measures. In the following paragraphs, these units are discussed in detail and modified to be used by RICH.

Category Utility measures the overall quality of a partition of instances into clusters. Consider a partition, $C = \{C_k\}$ where $1 \leq k \leq n$, found by a clustering algorithm based on given attributes A_i, where $1 \leq i \leq m$. All attributes are categorical or nominal and have a set of attribute values of categories, $\{V_{ij}\}$. The category utility function scores partition C against the set of variables according to the following definition:

$$CU(C) = \frac{1}{n}\sum_{k=1}^{n} P(C_k)\left[\sum_{i=1}^{m}\sum_{j} P(A_i = V_{ij} \mid C_k)^2 - \sum_{i=1}^{m}\sum_{j} P(A_i = V_{ij})^2\right] \qquad (8)$$

The differences between summed squares of probabilities gives the increase in the expected number of attribute values that can be predicted given a class, C_k, over the expected number of attribute values that could be predicted without using the class [52]. The difference is then summed over all clusters weighted by their sizes $P(C_k)$ in

the outer summation. The division by n takes into account different partition sizes. Higher the CU value better is the clustering.

The term between the square brackets can be re-written as follows:

$$\sum_{i=1}^{m}\sum_{j}\left[P(A_i = V_{ij} \mid C_k)^2 - P(A_i = V_{ij})^2\right] \tag{9}$$

For RICH, the difference in squares of probabilities

$$P(A_i = V_{ij} \mid C_k)^2 - P(A_i = V_{ij})^2 \tag{10}$$

will almost always be very low (or negative) unless more frequent items were considered than those actually belonging to the cluster. The negative result in summation is due to the fact that when RICH clusters attributes, it considers only those attributes that are frequent. Therefore this unit cannot be used to evaluating attribute clustering.

Another popular evaluation unit is Significance Test on External Variable, as used by several categorical clustering algorithms [23] [53]. This technique evaluates the quality of a cluster based on an external attribute not used for clustering. Therefore entropy of the clustering solution is calculated using a variable, for example a class attribute, that did not participate in cluster generation. The entropy of an attribute A_i in a cluster C_k where A_i can take a value denoted by V_j, is given by the following equation:

$$E(C_k) = \sum_{j} P(A_i = V_j) \log P(A_i = V_j) \tag{11}$$

This unit again, like CU, is unjust when evaluating the clusters generated by RICH, as it requires all attributes to belong to certain clusters, whereas RICH chooses only those attributes for representation of a cluster, which are frequent enough.

Other metrics like precision, recall can be applied for clustering evaluation in same essence, but will fail to correctly evaluate the attribute clusters generated by RICH. Therefore we present an evaluation metric that can be used for computing quality of item clusters generated from a pre-classified multi-relational dataset.

Let $C = \{C_1, C_2, \ldots, C_k\}$ be a set of item clusters generated from a database D consisting of the relations $\Re = \{R_1, R_2, \ldots, R_m\}$. Each cluster C_k is then a set of items of the form $R_m.A_i = V_j$ where $1 \leq i \leq$ number of attributes in a relation R and $A_i \, \varepsilon \, A$, the set of attributes in a relation R_m and V_j is one of the values that A_i can take. $P = \{P_1, P_2, \ldots, P_k\}$ is a set of class labels with which each tuple in the target relation $\Gamma \, \varepsilon \, \Re$, the relation having the class attribute Λ, is labeled. There is a one to one correspondence between C_k and P_k, i.e. each cluster C_k a set of attribute-value pairs or items that best represent tuples belonging to the class P_k.

Then the cluster accuracy φ_k for a cluster C_k is defined as the ratio of the number of tuples covered by the cluster for a particular class P_k to the number of tuples in that class:

$$\varphi_k = \frac{\|T_k\|}{\|\Gamma.\Lambda = P_k\|} \tag{12}$$

T_k is the set of tuples in Γ that satisfy the attribute-value pairs defined by a cluster C_k. As these attribute-value pairs can belong to any table R_k, the tables need to be joined using a common attribute Q. As it is not known which cluster represents which class, we find T_k for each class value P_k and chose the best one for calculating φ_k. T_k is calculated using the formula given below:

$$T_k = \left\{ t \mid t = \bigcap_{l=1}^{|C_k|} \left[R_{ml}.A_{il} = V_{jl} \bigcap \Gamma.\Lambda = P_k \bigcap R_{ml}.Q = \Gamma.Q \right] \right\} \tag{13}$$

φ is able to capture the information in the clusters and map it onto the relations so that the quality of each cluster is evaluated. φ has a value between 0 and 1.0, 0 being the lowest value φ can have indicating poor cluster accuracy and 1.0 indicating high cluster accuracy. φ is used to evaluate cluster accuracy of clusters generated by RICH. These experiments are discussed in the next sub-section. Please note, from here onwards the terms 'accuracy' and 'quality' are used interchangeably to refer to φ.

6.3.2 Cluster Quality

Running RICH on Mushroom, Congressional voting, PKDD and CoMMA datasets generated item clusters shown in Table 20. Figure 28, Figure 29 and Figure 30 show the accuracy φ of the clusters generated with respect to support. Single table datasets are compared against support while PKDD is compared against cross support.

Cluster quality is measured using the cluster accuracy metric discussed in the previous sub-section. The CoMMA dataset is not pre-classified and therefore the cluster accuracy metric cannot be applied to it. However the results obtained by clustering have been included in Table 20 for readers to gauge the clustering capacity of RICH. The CoMMA dataset is a multimedia dataset, with images belonging to nature domain. There are two clusters generated from it using RICH, one of mountains, snow and lakes and the other of trees, bushes and houses, which clearly indicates good clustering.

Table 20: Item Clusters generated by RICH for different Datasets

Dataset	Cluster	Items in Cluster
Mushroom	Edible	ring_number=one, gill_attachment=free, veil_color=white, stalk_surface_above_ring=smooth, gill_size=broad, gill_spacing=close
	Poisonous	stalk_shape=tapering, veil_type=partial, bruises?=no, stalk_surface_below_ring=smooth
Congressional Voting	Republican	el_salvador_aid='Y', crime_prevention='Y', religious_groups_in_schools='Y', duty_free_exports='N', handicapped_infants='N'
	Democrat	mx_missile='Y', aid_to_nicaraguan_contras='Y', el_salvador_aid='N', physician_fee_freeze='N', education_spending='N', adoption_of_the_budget_resolution='Y', anti_satellite_test_ban='Y'
PKDD '99	Loan Status=1	ordertab.bank_to='KL', account.district_id='59', account.frequency='MONTHLY', ordertab.account_to='67186093', ordertab.K_symbol='LOAN', account.date='970408'
	Loan Status=2	ordertab.bank_to='MN', ordertab.account_to='86397230', account.district_id='31', ordertab.amount='13386', ordertab.account_to='14132368', loan.duration='12', loan.date='960429', loan.amount='30276', loan.payments='2523', account.date='950407'
CoMMA	Cluster 0	annotations: GLACIER SNOW GREEN MOUNTAINS LAKE BLUE WHITE CLOUDS RIVER SNOW-CLAD SETTLEMENT imagefeatures: 463B 19D0 8D45 21D135 13D90 12I 7I 10D45 8D90 35Y 13D0 1D45 1D135 12D45 12D135 22D0 10D90 6I 9D90 8D135 4Y 13D45 12D0 35D0 14D45 13I 24D90 30D135 15D135 26I 15D90 32D0 12D90 31D0 15D45 10D135 14D135 8I 13D135 457B 9D45 9D135
	Cluster 1	annotations: BUSHES SKY HOUSE CHERRY OVERCAST TREES WATER GRASS SIDEWALK BUILDINGS LEAFLESS GROUND CLOUDY SEA ISLAND STREET TREE BUILDING CLEAR WAVES TIDE SHORE imagefeatures: 0Y 5D45 6D135 0G 8D0 10I 0R 7D135 0B 1I 11D0 6D45 2I 2D45 2D90 2D135 4D45 3D90 5D0 11D45 7D90 28D90 4D90 3D45 5D90 6D90 7D0 3D135 4D0 4I 9I 5I 10D0 34D45 17D90 4D135 6D0

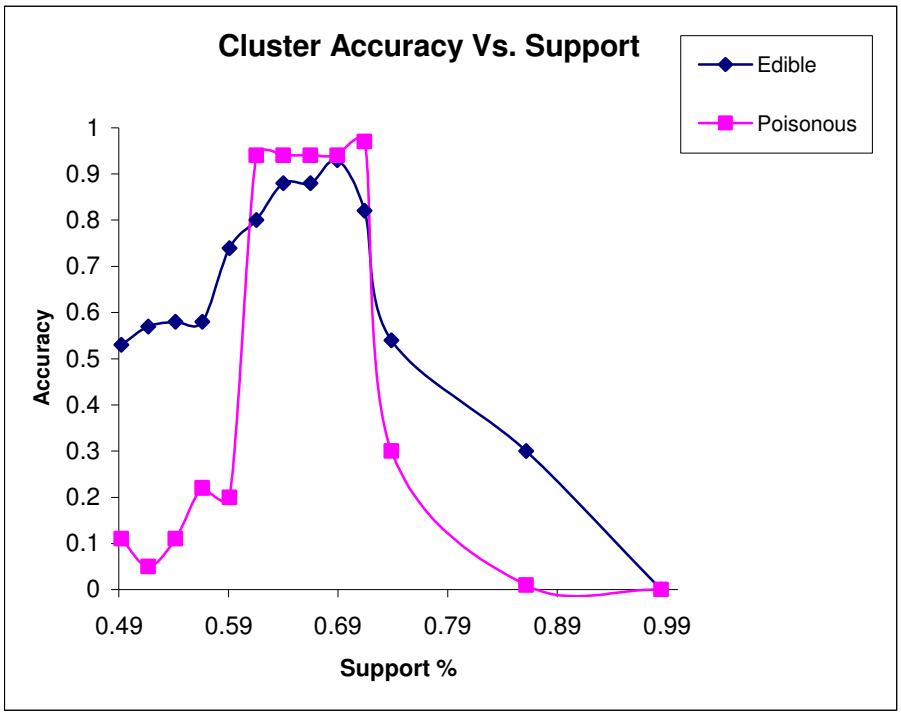

Figure 28: Cluster Accuracy Vs Support for Mushroom Dataset

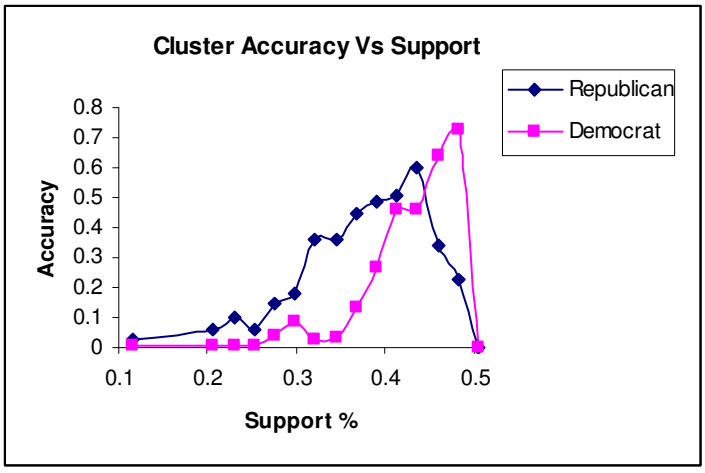

Figure 29: Cluster Accuracy Vs Support for Congressional Voting Dataset

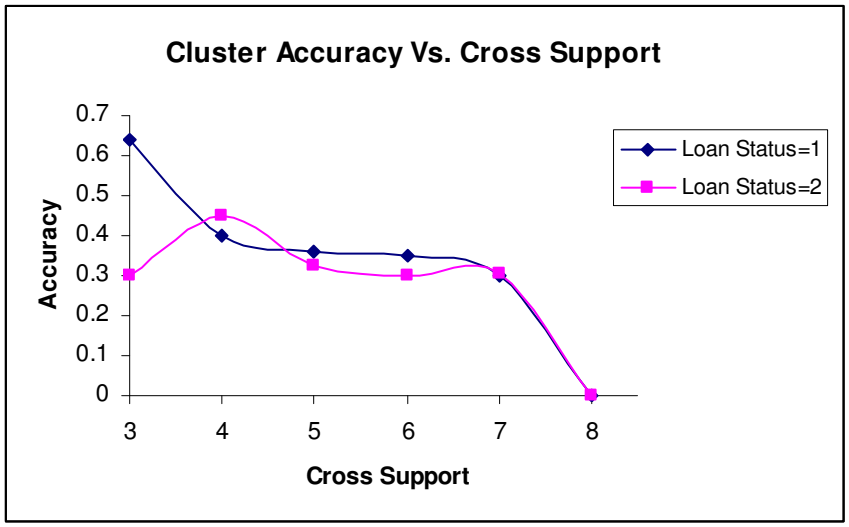

Figure 30: Cluster Accuracy Vs Cross Support for PKDD dataset

As can be seen from the graphs given above, the cluster accuracy φ is initially low for a dataset; it then improves as support increases and again dips down. This behavior of cluster accuracy is expected, as initially when support is low, rules generated include the 'not-so-frequent-items'. Gradually as support is increased, the cluster definitions become stricter and are able to correctly label tuples to their respective classes. After a particular support limit, there is a sharp drop in the number of items in a cluster (which eventually becomes 0 if not many items are as frequent as the support), and therefore the accuracy degrades and finally becomes 0. Please note that for all the experiments to measure cluster accuracy, the class label attribute was *not* included in the clustering process; as that would generate incorrect results.

Table 21: Cluster Accuracy (φ) Comparison with CLUTO on Congressional Voting

	Republican	**Democrat**
RICH	0.23	0.73
CLUTO	0.00	0.84

Table 22: Cluster Accuracy (φ) Comparison with CLUTO on Mushroom Dataset

	Poisonous	**Edible**
RICH	0.93	0.94
CLUTO	0.51	0.81

Cluster accuracy as compared to CLUTO is given in Table 21 for Voting dataset and Table 22 for the Mushroom dataset. As CLUTO is a single table clustering algorithm, its performance on multi-relational datasets cannot be tested. The best cluster accuracy for a single RICH run is used for comparison with CLUTO. RICH scores over CLUTO with a big margin on the Mushroom dataset by correctly identifying 45% more records as poisonous and 13% more records as edible. On the other hand, CLUTO generates good cluster for Democrats in the Voting dataset, but fails to identify any of the Republican records correctly.

6.3.3 RICH Runtime Performance

Runtime performance of RICH for different datasets is show below.

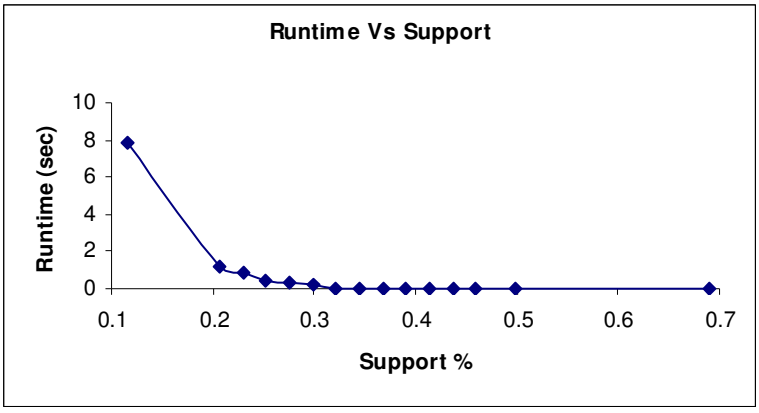

Figure 31: Runtime Vs Support for Congressional Voting Dataset

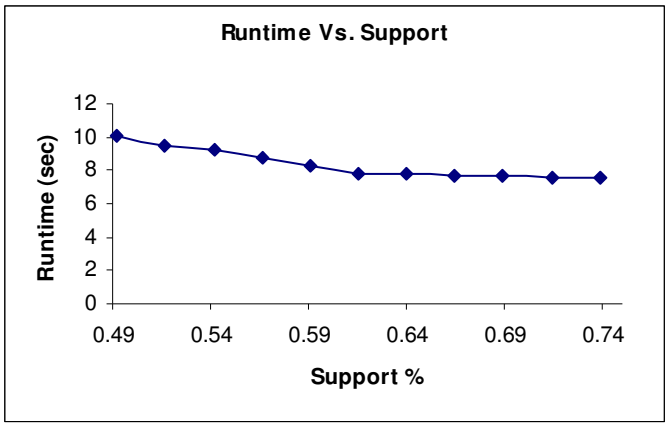

Figure 32: Runtime Vs Support for Mushroom Dataset

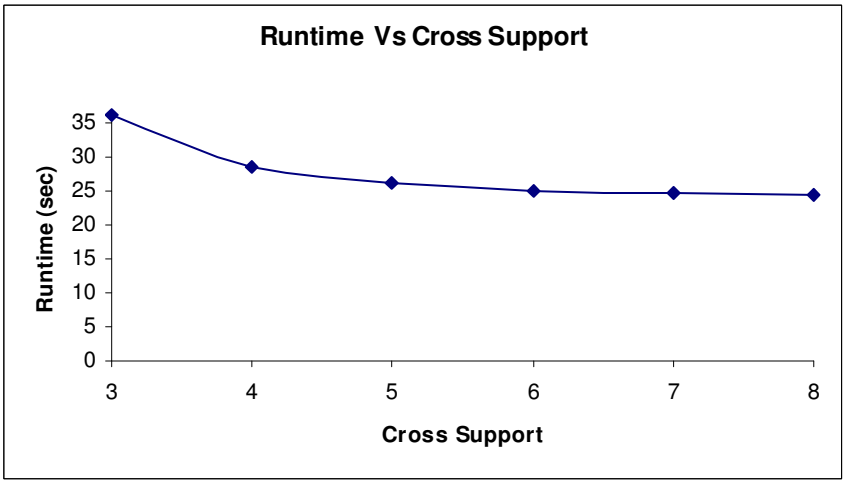

Figure 33: Runtime Vs Cross Support for PKDD Dataset

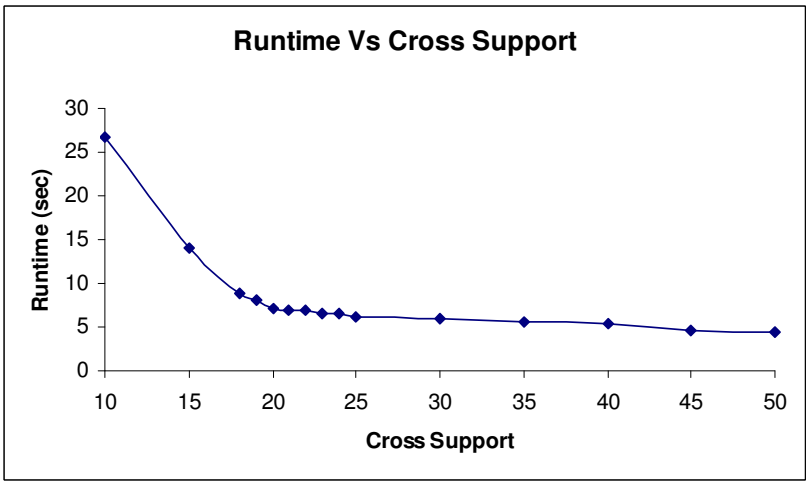

Figure 34: Runtime Vs Cross Support for CoMMA dataset

RICH, as can be inferred from the graphs, takes longer time for smaller values of support. The reason being, with small support values, more items qualify for frequent patterns, therefore the MRFP Trees generated for the tables are taller and mining these tall trees for frequent patterns take longer. With increase in support, fewer items are ‘frequent-enough’ and thus mining the trees takes lesser time, and the runtime improves.

7. Conclusion

The primary objective of this thesis was to explore the realm of multi-relational data mining and propose algorithms for multi-relational frequent pattern mining and clustering. MRFP Growth is an algorithm for frequent itemset mining in multi-relational data. The ID propagation technique used by MRFP Growth enables multiple relations to be joined via the frequent itemsets found in each relation. The initial stage of concurrent and independent relation processing ensures faster execution time and scalability. The performance of MRFP Growth was effective as shown by experimental results. It scored over WARMR and Decentralized Apriori in terms of number of frequent patterns generated and execution time.

The RICH approach to clustering multi-relational data was presented in the thesis. RICH exploits the frequent patterns generated by MRFP Growth to make clusters of itemsets using a hypergraph model. Accuracy of the item clusters thus generated was tested on pre-classified multi-relational and single-relation data. The results have shown that RICH is a competitive multi-relational item clustering approach.

The thesis also presents an implementation of MRFP Growth for auto annotation of images. This work is called CoMMA - Combining Multi-relational Multimedia for Associations. The framework architecture, its application as an image annotation generator, the experiments and results to evaluate CoMMA's performance were presented in detail.

Possible future work that can extend the work presented in this thesis includes:

MRFP Growth: Making MRFP Growth a part memory-part disk based algorithm and exploring closed itemset mining using MRFP Growth. Optimizing MRFP Growth for better performance. Automating pattern generation process using MRFP Growth on relational data with different types of joins.

RICH: an application of RICH to real world problems. Using confidence for cluster generation. Comparing RICH's performance with a multi-relational item clustering algorithm

CoMMA: Expanding CoMMA's domain to other than nature.

8. Bibliography

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. *In Proc. of the ACM SIGMOD Conference on Management of Data*, pages 207-216, Washington, D.C., May 1993.

[2] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A.I. Verkamo. *Fast Discovery of Association Rules*, in U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy (eds.), *Advances in Knowledge Discovery and Data Mining,* chapter 12, pages 307-328, AAAI/MIT Press, 1996.

[3] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. *In Proc. of the ACM SIGMOD Conference on Management of Data*, Dallas, 2000.

[4] J. Han, M. Kamber. In J. Gray (ed), *Data Mining: Concepts and Techniques*, Academic Press, 2001.

[5] S. Džeroski. *Multi-Relational Data Mining: An Introduction.* ACM SIGKDD Explorations Newsletter, Volume 5, Issue 1, Pages 1-16, 2003.

[6] S. Muggleton (ed). *Inductive Logic Programming*, Academic Press, 1992.

[7] L. Dehaspe and H. Toivonen. *Discovery of Frequent Datalog Patterns*. Data Mining and Knowledge Discovery, 3(1):7-36, 1999.

[8] V. C. Jensen and N. Soparkar. Frequent Itemset Counting Across Multiple tables. *In Proc. of the PAKDD*, pages 49-61, 2000.

[9] E-H. Han, G. Karypis, V. Kumar, B. Mobasher. *Hypergraph Based Clustering in High-Dimensional Data Sets: A Summary of Results*. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 1997.

[10] C. Berge. *Graphs and Hypergraphs*. American Elsevier, 1976.

[11] G. Karypis. *CLUTO – A Clustering Toolkit.* Department of Computer Science, University of Minnesota, 2002.

[12] J. Han, Data Mining, in J. Urban and P. Dasgupta (eds.). *Encyclopedia of Distributed Computing*, Kluwer Academic Publishers, 1999.

[13] H. Garcia-Molina, J. D. Ullman, and J. Widom. *Database Systems: The Complete Book*, Prentice Hall, 2002.

[14] C. Nedellec, C. Rouveirol, H. Ade, F. Bergadano, and B. Tausend. *Declarative bias in inductive logic programming*. In L. De Raedt (ed), Advances in Inductive Logic Programming, pages 82-103, IOS Press, 1996.

[15] R. Iváncsy, F. Kovács and I. Vajk. *An Analysis of Association Rule Mining Algorithms.* In the Fourth International ICSC Symposium on Engineering of Intelligent Systems (EIS 2004), 2004.

[16] G. Plotkin. *A note on inductive generalization*. In B. Meltzer and D. Michie (eds.), Machine Intelligence 5, pages 153-163. Edinburgh University Press, Edinburgh, 1969.

[17] J-U. Kietz and M. L¨ubbe. An efficient subsumption algorithm for inductive logic programming. In S. Wrobel (ed), *Proc. of the 4th International Workshop on Inductive Logic Programming*, volume 237, pages 97–106. Gesellschaft für Mathematik und Datenverarbeitung MBH, 1994.

[18] W. Emde and D. Wettschereck. Relational instance based learning. *In Proc. of the Thirteenth International Conference on Machine Learning*, pages 122-130. Morgan Kaufmann, San Mateo, CA, 1996.

[19] M. Kirsten, S. Wrobel, and T. Horvith. *Distance Based Approaches to Relational Learning and Clustering*. In [5], pages 213-232, 2001.

[20] P. Berkhin. *Survey of Clustering Data Mining Techniques*. Accrue Software Inc., San Jose, CA.

[21] P. Sneath and R.Sokal. Numerical Taxonomy. W. H. Freeman, San Francisco, 1973.

[22] D. Gibson, J. Kleinberg, and P. Raghavan. Clustering categorical data: An approach based on dynamical systems. *In Proc. of the 24th International Conference on Very Large Databases*, pages 311-323, New York City, August 24-27 1988.

[23] D. Barbará, J. Cluto, and Y. Li. *COOLCAT: An entropy-based algorithm for categorical clustering*. In CIKM, McLean, VA, 2002.

[24] V. Ganti, J. Gehrke, and R. Ramakrishnan. Cactus: clustering categorical data using summaries. *In Proc. of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 73–83. ACM Press, 1999.

[25] I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. *In Proc. of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 89–98. ACM Press, 2003.

[26] G. Das, H. Mannila, and P. Ronkainen. *Similarity of attributes by external probes*. In Knowledge Discovery and Data Mining, pages 23–29, 1998.

[27] G. Das, H. Mannila, and P. Ronkainen. *Context based similarity measures for categorical databases*. In Principles and Practice of Knowledge Discovery in Databases, pages 201–210, September 2000.

[28] C. C. Aggarwal. Towards systematic design of distance functions for data mining applications. *In Proc. of the ninth ACM SIGKDD International Conference on Knowledge discovery and data mining*, pages 9–18. ACM Press, 2003.

[29] G. Karypis, R. Aggarwal, V. Kumar and S. Shekhar. Multilevel hypergraph partitioning Application in VLSI domain. *In Proc. of ACM_IEEE Design Automation Conference, 1997*.

[30] Beng Chin Ooi, *Viper Image Database System*, National University of Singapore.

[31] Osmar R. Zaiane, Jiawei Han, Ze-Nian Li, Sonny H. Chee, and Jenny Chiang. MultiMediaMiner: a system prototype for multimedia data mining. *In Proc.of*

ACM SIGMOD International Conference. on Management of Data, 581-583, 1998.

[32] K. Barnard et. al. *Matching Words and Pictures*. Journal of Machine Learning Research, 3(2003), 1107-1135

[33] C. Carson et al. Blobworld: a system for region-based image indexing and retrieval. *In Proc. Of the 3rd International Conference on Vision Information Systems*, Amsterdam, Netherlands, June 1999, pp. 509-516.

[34] P. Cheng and L. Chien. *Auto-Generation of Topic Hierarchies for Web Images from Users' Perspectives*. Conference on Information and Knowledge Management '03, November 3-8 2003, 544-547.

[35] J. Jeon and R. Manmatha. *Automatic Image Annotation and Retrieval using Cross-Media Relevance Models*. Special Interest Group on Information Retrieval'03, July 28-August 1, 2003.

[36] J. Dai, M. Li Lee and W. Hsu. Mining Viewpoint Patterns in Image Databases. *In Proc. of 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Washington, DC, USA, August 2003.

[37] P. Suetens, P. Fua, and A. J. Hanson. *Computational Strategies for Object Recognition*, ACM Computing Surveys, March 5 1992, 5-62.

[38] W. Hsu, M. Li Lee and J. Zhang. *Image Mining: Trends and Developments*. In Journal of Intelligent Information System (JISS): Special Issue on Multimedia Data Mining, Kluwer Academic, 2002.

[39] L. Itti, and C. Koch. *Computational modeling of Visual Attention*. Nature Neuroscience Review, 2(3):194-203. 2001.

[40] R. Gaborski, V.S. Vaingankar, and R.L. Canosa. Goal Directed Visual Search Based on Color Cues: Cooperative Effects of Top-down & Bottom-up Visual Attention. *In Proc. of the Artificial Neural Networks in Engineering*, Rolla, Missouri. Vol 13, pp: 613-618, 2003.

[41] M. Flickner et al., Query by Image and Video Content: The Cubic System, IEEE Computer, 28(9): 23-32, Sep. 1995.

[42] R. Zhang and Z. Zhang. Addressing CBIR Efficiency, Effectiveness and Retrieval Subjectivity Simultaneously. *In Proc. of the 5th ACM SIGMM international workshop on Multimedia information retrieval*, Nov. 7, 2003, Pages: 71-78.

[43] L. Wenyin, S. Dumais, Y. Sun, H. Zhang, M. Czerwinski and B. Field. *Semi-Automatic Image Annotation*. Microsoft Research Technical Report.

[44] Z. Su, S. Li and H. Zhang. Extraction of Feature Subspaces for Content Based Retrieval Using Relevance Feedback. *In Proc. of the ninth ACM international conference on Multimedia*, Sep 3- Oct 5, 2001, 98-106.

[45] F. Monay and D. Gatica-Perezl. On Image Auto-Annotation with Latent Space Models. *In Proc. of the ninth ACM international conference on Multimedia*, November 2-8, 2003, Pages: 275-278.

[46] L. Jia, and W. Z. James. *Automatic Linguistic Indexing of Pictures by a Statistical Modeling Approach*. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 25, No. 9, September 2003.

[47] A. Teredesai, J. Kanodia, Muhammad A., R. Gaborski, *CoMMA: A Framework for Multi-media Mining using Multi-relational Associations*, MDM KDD 2004.

[48] A. Teredesai, A. Muhammad, J. Kanodia, R. Gaborski, *CoMMA: A Framework for Multi-media Mining using Multi-relational Associations*, Knowledge and Information Systems, 2005.

[49] M. Bongard. *Pattern Recognition*. Hayden Book Company (Spartan Books), 1970.

[50] C.Blake(Librarian).UCI Machine Learning Repository. http://www.ics.uci.edu/mlearn/MLRepository.

[51] M. A. Gluck and J. E. Corter. Information, Uncertainty, and the Utility of Categories. In COGSCI, Irvine, CA, USA, 1985.

[52] B. Mirkin. *Reinterpreting the Category Utility Function*. In Douglas Fisher (ed.) Machine Learning, pages 1-11, Kluwer Academic Publishers, Boston.

[53] P. Andritsos, P. Tsaparas, R. J. Miller and K. C. Sevcik. LIMBO: Scalable Clustering of Categorical Data. *In Proc. of the 9th International Conference on Extending Database Technology*, March 2004.

[54] L. Parsons, E. Haque and H. Liu. *Subspace clustering for high dimensional data: a review*. ACM SIGKDD Explorations Newsletter, vol 6, pages 90-105, June 2004.

[55] C. Baumgartner, C. Plant, K. Kailing, H. Kriegel and P. Kroger. Subspace Selection for Clustering High-Dimensional data. *In Proc. of the Fourth IEEE International Conference of Data Mining*, Brighton, UK, 2004.

[56] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. *In Proceedings of the 1998 SIGMOD Conference*, Seattle, Washington, 1998.