DATA MINING FOR RULE DISCOVERY IN RELATIONAL DATABASES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

SERKAN TOPRAK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2004

Approval of the Graduate School of Natural and Applied Sciences.

Prof. Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Ayşe Kiper
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Ferda N. Alpaslan
Supervisor

Examining Committee Members

Prof. Dr. Adnan Yazıcı (METU, CENG) _____

Assoc. Prof. Dr. Ferda N. Alpaslan (METU, CENG) _____

Prof. Dr. Mehmet R. Tolun (Çankaya University) _____

Assoc. Prof. Dr. İsmail H. Toroslu (METU, CENG) _____

Dr. Ayşenur Birtürk (METU, CENG) _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name:

Signature :

ABSTRACT

DATA MINING FOR RULE DISCOVERY IN RELATIONAL DATABASES

Toprak, Serkan

M.Sc., Department of Computer Engineering

Supervisor: Assoc. Prof. Dr. Ferda N. Alpaslan

September 2004, 53 pages

Data is mostly stored in relational databases today. However, most data mining algorithms are not capable of working on data stored in relational databases directly. Instead they require a preprocessing step for transforming relational data into algorithm specified form. Moreover, several data mining algorithms provide solutions for single relations only. Therefore, valuable hidden knowledge involving multiple relations remains undiscovered. In this thesis, an implementation is developed for discovering multi-relational association rules in relational databases. The implementation is based on a framework providing a representation of patterns in relational databases, refinement methods of patterns, and primitives for obtaining necessary record counts from database to calculate measures for patterns. The framework exploits meta-data of relational databases for pruning search space of patterns. The implementation extends the framework by employing Apriori algorithm for further pruning the search space and discovering relational recursive patterns. Apriori algorithm is used for finding large itemsets of tables, which are used to refine patterns. Apriori algorithm

is modified by changing support calculation method for itemsets. A method for determining recursive relations is described and a solution is provided for handling recursive patterns using aliases. Additionally, continuous attributes of tables are discretized utilizing equal-depth partitioning. The implementation is tested with gene localization prediction task of KDD Cup 2001 and results are compared to those of the winner approach.

Keywords: Relational Data Mining, Association Rules, Relational Databases, Apriori Algorithm, Discretization, Recursive Relations

ÖZ

İLİŞKİSEL VERİ TABANLARINDA VERİ MADENCİLİĞİ İLE KURALLAR BULUNMASI

Toprak, Serkan

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi: Doç. Dr. Ferda Nur Alpaslan

Eylül 2004, 53 sayfa

Günümüzde veri genellikle ilişkisel veri tabanlarında saklanmaktadır. Buna rağmen çoğu veri madenciliği algoritması ilişkisel veri tabanlarındaki veri ile doğrudan çalışamamaktadır. Dolayısıyla bu algoritmaları ilişkisel veri üzerinde çalıştırabilmek için ilişkisel veriyi ön işlemeyle algoritmaların anlayacağı biçime çevirmek gerekmektedir. Ayrıca birçok veri madenciliği algoritması tek ilişki kullanan çözümler sunmaktadır. Dolayısıyla çoklu ilişkisel yapıda olan gizli kalmış değerli bir bilgiyi bu algoritmaları kullanarak elde etmek mümkün olmamaktadır. Bu çalışma kapsamında ilişkisel veri tabanları üzerinde çoklu ilişkisel yapıdaki ortak kuralları bulmayı sağlayan bir uygulama geliştirilmiştir. Uygulama altyapısı olarak ilişkisel veri tabanlarındaki desenleri tanımlayabilen, bu desenleri eklerle geliştirebilen ve bu desenlerin çeşitli ölçmeleri için gerekli sayımları veri tabanından temel yetilerle alan bir yapı kullanılmıştır. Bu altyapı, veri tabanının tanımında yer alan bilgileri kullanarak arama alanının daraltılmasını sağlamıştır. Bu çalışma, Apriori algoritmasını arama alanını daha da küçültmek

için kullanarak ve altyapı tarafından desteklenmeyen özyinelemeli desenlerin bulunmasını sağlayarak altyapıya yenilikler getirmiştir. Apriori algoritması her tablo üzerinde sık karşılaşılan desenleri bulmak için kullanılmış ve bu algoritmanın gerekli destek değerini bulma yöntemi değiştirilmiştir. Veri tabanındaki özyinelemeli ilişkileri belirlemek için bir yöntem sunulmuş ve uygulama bu durumlar için tablo kısaltmalarının kullanıldığı bir çözüm sağlamıştır. Veri tabanı alanlarında saklanan sürekli değerleri bölümleyebilmek için eşit derinlik yöntemi kullanılmıştır. Uygulama bir veri madenciliği yarışması olan KDD Cup 2001'den alınan örnek bir genlerde yer tahmini problemi ile test edilmiş ve ortaya çıkan sonuçlar yarışmayı kazanan yaklaşımın sonuçlarıyla karşılaştırılmıştır.

Anahtar Kelimeler: İlişkisel Veri Madenciliği, Ortak Kurallar, İlişkisel Veri Tabanları, Apriori Algoritması, Bölümleme, Özyinelemeli İlişkiler

To Seda *with love. . . .*

ACKNOWLEDGMENTS

I would like to thank my supervisor, Assoc. Prof. Dr. Ferda Nur Alpaslan, who encouraged and guided me in writing of this thesis. I also express my sincere appreciation to Assoc. Prof. Dr. İ. Hakkı Toroslu, who also guided me in this study. Thanks to the other committee members, Prof. Dr. Mehmet R. Tolun, Prof. Dr. Adnan Yazıcı and Dr. Ayşenur Birtürk for their comments and suggestions.

I would like to express my deepest gratitude to my wife, Seda, who encouraged and supported me in this demanding study with her never ending patience.

I would also like to thank to my employer, MilSOFT, for providing me time for studying whenever I needed.

Thanks are also to all of my friends for their direct or indirect help. Finally, my deepest thanks are to my parents who supported and motivated me with their never ending patience, tolerance and understanding throughout the study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Rationale

Large amounts of information have been gathered from several fields including sales, marketing, chemistry, biology, banking, and have been stored in databases all around the world. This information is mostly used and maintained by an application performing regular tasks such as providing personnel list or storing the transactions of a store. However stored information can also provide some valuable knowledge that could help the owners of the information to improve their benefit. For example a GSM company knowing that "people between ages 18 and 25 tends to make fewer calls during weekends" can start a campaign like %50 discount for teenagers on weekends to favor calls of that age group at weekends. The process of extracting such valuable knowledge from a database is known as Data Mining.

Data mining is often defined as finding hidden knowledge in a given set of data. There are various data mining tasks and various algorithms for solving them. Most of data mining algorithms run on single table databases or single relations. However, hidden knowledge to be found may be related to multiple tables or multiple relations. Data mining over multiple tables or multiple relations is known as multi-relational data mining. Relations can be in the form of logic

programs or in the form of tables in a relational database. Since data is mostly stored in relational databases, data mining algorithms directly working with relational data stored in relational databases are practical, at least they do not need data to be transformed into a different format required by the algorithm.

Each data mining algorithm have a searching step in which data is searched to discover valuable hidden knowledge for the given task. When upgrading from single relation data mining to multi-relational data mining, search space increases drastically. Therefore, a multi-relational data mining algorithm should have methods for enormous search space. Otherwise, performance of the algorithm would be very low.

There are various data mining algorithms directly working with data stored in relational databases for data mining tasks including classification, regression and association rule discovery. Some of them require modification of existing tables in database by adding columns to be used by the algorithm. Some of them require creation of new tables in database for storing temporary information needed by the algorithm. Modification of database can be infeasible in some cases. For example in a large database, creation of new tables for storing the records already processed would create a table with the same size as original table, or users running data mining algorithm may not have the right to add new columns or create new tables. Therefore, it is better to have a data mining algorithm requiring no modifications on relational databases.

Sometimes, hidden knowledge to be discovered in a relational database may have a recursive description. For example, description of a relation R can include relation R itself. It is important for a data mining algorithm to find recursive descriptions of relations in a relational database.

In this study, a multi-relational data mining algorithm for discovering rules in a relational database is implemented providing solutions to difficulties presented so far. The algorithm employs methods to prune search space, can discover recursive rule descriptions and requires no modifications to given relational database.

1.2 Approach

In this thesis, data mining task of discovering association rules from a relational database is implemented based on the framework proposed in [1].

This framework employs defined foreign key constraints of a relational database to prune the search space. In addition, a measure providing the likelihood of refining a candidate description to yield valuable knowledge is used. This measure is employed in to eliminate weak candidates for pruning the search space.

Implementation presented in this study works on relational databases without requiring the modification of existing tables or introducing new tables. Relational database is only accessed for running "select" queries. Therefore, there is no extra storage needed for creating new tables and a user having only a "select" privilege on database can run this implementation for to discover rules.

Framework proposed in [1] excludes discovery of recursive association rules. However, this study extends this framework to discover recursive association rules. Conditions to determine whether a database has recursive associations are provided and aliases are used for tables so that a description can have one or more aliases of the same table.

In the implementation, a graph is constructed presenting the foreign key relations and aliases for tables. Aliases of tables form nodes and associations between aliases form edges in this graph. Several aliases may exist for the same table in the graph in case of recursive relationships. This graph is used during running Apriori algorithm [2]for each table and searching. An algorithm is implemented to find distinct paths with no repeating nodes between two aliases in the graph.

Equal-depth partitioning method is used in this study for partitioning continuous attributes of tables into discrete intervals. Partitions are used during running Apriori algorithm on tables.

Apriori algorithm is implemented to find a set of attribute=value conditions of each table with a support value greater than a specified support value. Apriori algorithm is modified by calculating support against the same table, target table,

for each table in this study. Results of Apriori algorithm is used in searching step.

In general, the framework in [1] is proposed to be used for all kind of data mining algorithms working on data stored in relational databases. Implementing WARMR, which is an extension of Apriori for mining association rules in multiple relations [3], is proposed as a sample instance of the framework. In this study, a software based on this framework is implemented to find association rules. By finding association rules describing a single attribute of a table, a set of rules classifying this attribute can be discovered. In this way, the implementation can be used for classification tasks also.

Developed software is tested with *Genes* data set of KDD Cup 2001 competition [4], and results of the tests are compared against the test data set. Tasks in this competition require multi-relational rules to be discovered on given data, which makes this dataset highly suitable for this study.

1.3 Road Map

Chapter 2 provides background for Knowledge Discovery in Databases (KDD), Data Mining, models and steps of KDD and Data Mining tasks. Then relational data mining approaches including Inductive Logic Programming (ILP), Bayesian Networks, Neural Networks and relational data mining on databases is presented.

Chapter 3 presents the framework used in this study. Pruning search space using foreign key constraints between tables, concepts of the framework, structures for representing relational rules and primitives for evaluating these relational rules are presented in this chapter.

Chapter 4 provides detailed information for the implementation of a rule discovery software for relational databases based on the framework presented in Chapter 3. Steps of Preprocessing of relational database information, discretization of continuous attributes, running Apriori algorithm on tables and searching for rules is described in this chapter. A solution for handling recursive associations is also described in this chapter.

Chapter 5 presents the testing approach, discusses the results of the tests and compares test results with the results of winning approach.

Chapter 6 concludes the study by providing a summary of study and test results. Limitations and further improvements of the study are presented in this chapter.

CHAPTER 2

DATA MINING BACKGROUND

This chapter provides background for Data Mining step of KDD process and describes the distinction between KDD and Data Mining. Types of data mining problems, such as classification and regression, and solutions to these problems are presented. Moreover relational data mining is described and approaches to relational data mining are discussed in this chapter.

2.1 Knowledge Discovery in Databases (KDD)

KDD is defined as the nontrivial extraction of implicit, previously unknown, and potentially useful information from data [5]. [6] defines KDD as the nontrivial process of identifying valid, novel, potentially useful and ultimately understandable patterns in data. This process consists of five steps; selection, preprocessing, transformation, data mining and interpretation/evaluation [7].

Selection is the task of learning the application domain and creating a target dataset by obtaining necessary data from several, possibly heterogeneous, data sources. Preprocessing is the task of removing anomalies (e.g. noise or outliers) in the data produced in selection step and deciding on strategies like handling the missing data fields. Transformation is the task of transforming the preprocessed data into a form with reduced number of features and variables under

consideration using dimensionality reduction and transformation methods for eliminating irrelevant features and variables. In data mining step, an algorithm is chosen according to purpose of problem, and this algorithm is applied to transformed data to find interesting hidden information in the data. Data mining step includes a searching mechanism to find interesting hidden patterns in transformed data. Finally, interpretation/evaluation is the last step of KDD process in which discovered patterns are interpreted, redundant and irrelevant patterns are removed and useful ones are presented in an human understandable format.

KDD is used in many application areas for discovery of valuable hidden knowledge out of stored data. [8] provides an overview of common knowledge discovery tasks and approaches to solve those tasks.

Some of main application areas of KDD are:

- Marketing [9, 10, 11, 12]
- Intrusion Detection[13, 14, 15, 16]
- Spatial Information Mining [17, 18, 19, 20, 21, 22]
- Web Mining[23, 24, 25, 26]
- Text Mining[27, 28, 29]
- Medical Information Mining[30, 31, 32, 33]

2.2 Data Mining

Most data mining methods are based on tried and tested techniques from three different fields: machine learning, pattern recognition and statistics [34]. However, those methods have a property in common. Most data mining algorithms can be viewed as compositions of a few basic techniques and principles; the model, the preference criterion and the search algorithm [7].

Model is defined as the language used to describe discoverable patterns [6]. If the representation is too limited, then no amount of training time or examples

can produce an accurate model for the data . Preference criteria are quantitative statements functions) of how well a particular pattern (a model and its parameters) meets the goals of the KDD process. Once the model and the preference criteria are fixed, then the data-mining problem has been reduced to purely an optimization task: Find the parameters and models from the selected family that optimize the evaluation criteria, that is searching.

Data mining models can be divided into two categories, Predictive and Descriptive models. Predictive models try to predict a value for the result of an unknown case using the known results of some existing data. On the other hand, descriptive algorithms try to find hidden patterns or relations in data. Classification, regression and time series analysis are the major predictive data mining algorithms, whereas clustering, summarization, association rule extraction and sequence discovery algorithms are the major descriptive algorithms.

Classification algorithms predict the class of an instance using the given values for other attributes of that instance. Classes are determined prior to the data mining task by the user, therefore classification is a supervised algorithm. Using the data given, classification rules are found, which are applied to incoming instance to predict which class it belongs. For example a classification algorithm can be used to predict whether the customer of a bank can pay back the credit s/he borrows using the past credit records of other customers.

Regression algorithms predict the value of a continuous attribute of an instance using the given values for other attributes. Using the data given, a function is found such that given the values of other attributes, it returns the value of the continuous attribute. Using this function, missing continuous value of an incoming instance can be predicted. For example a regression algorithm can be used to predict a safe credit card limit value for a customer.

Time series analysis algorithms use temporal data to predict the future value or class of an instance. For example, stock value of a company on two days later can be predicted by using the stock values of the company for past 2 weeks and finding a similarity of these values with other companies' values which have

shown similar trends in the past.

Clustering algorithms describe the given data by grouping them into clusters determined by the algorithm itself. The difference between clustering and classification tasks is that the clustering task is unsupervised as the clusters are determined by the algorithm. For example, given a set of images, a clustering algorithm can form groups like forest, city, and beach, and then place the related images into appropriate clusters. There would be no group for beach category, if there was not an image related to beach.

Summarization algorithms try to find compact descriptions for the mined data.

Association rule extraction algorithms try to find some relationships or hidden patterns in data. Given data, these algorithms try to find rules in if-then form [2]. For example, given a bookstore's sales database, such an algorithm can produce a rule saying "If a customer buys a fantasy book, s/he also buys a mythology book".

Sequential discovery algorithms try to find patterns that are frequent in the given sequential data. Sequential discovery algorithms and time series analysis algorithms are similar for they both operate on data having a time dimension. The difference is that sequential discovery algorithms look for patterns whereas time series analysis algorithms predict a future value for an instance. An example to a sequential pattern would be a rule like "If a customer buys a fantasy book, s/he will also buy a mythology book in a month."

Large search space size, I/O operations and methods needed to retrieve data where it is stored are the major contributors to the complexity of data mining algorithms. Relational Database Management Systems (RDBMS) offer efficient indexing and retrieval of stored data. Therefore, RDBMS have been considered for efficient data mining to improve the efficiency of data retrieval and I/O operations. Using parallel database servers and mapping KDD primitives such as evaluation of a candidate rule into parallel database servers are proposed in [35] for speeding up knowledge discovery. [36] presents the discovery of asso-

ciation rules by using functionalities of a database management system with an efficiency comparable to specialized techniques. Even approaches combining advantageous sides of both Inductive Logic Programming (ILP) learning and relational databases exist [37]. ILP has the capability of representing first order rules which makes it more expressive than relational database's simple language representation using SQL. However algorithms employing the simple representation, database primitives and functions are proven to be efficient. Therefore, the proposed approach in [37] combines both approaches and uses relational database algorithms for smaller data mining tasks and then constructs the result by applying ILP methods on the results of smaller tasks.

2.3 Relational Data Mining

Most data mining algorithms operate on a single table. However most of the real world databases are relational and interesting patterns to be discovered can be scattered across multiple tables or relations. For example, daughter pattern can be found considering two relations, female and parent. Therefore Relational Data Mining (RDM) algorithms have been emerged for both predictive and descriptive methods. To emphasize the fact that RDM involves multiple tables (relations), RDM is often referred to as Multi-Relational Data Mining (MRDM)[38]. There are four approaches for multi-relational data mining; Inductive Logic Programming (ILP), Bayesian Networks, Neural Networks and Multi-Relational Data Mining on relational databases.

2.3.1 Inductive Logic Programming (ILP)

Inductive Logic Programming is formed at the intersection of Logic Programming and Machine Learning. ILP involves the use of background knowledge to construct an hypothesis which agrees with some set of observations according to given relations [39]. Propositional systems find rules involving one relation, while ILP systems can also find rules involving several relations. Using propositional approach, problems involving several relations can sometimes be solved

by calculating a universal relation (by joining all the relations into one relation), however this a costly operation and universal join may be very large to handle [40]. Therefore, ILP is a preferred choice for multi-relational data mining over propositional approaches.

ILP algorithms operate on data which is in the form of programming language clauses. It is also possible to link ILP systems to relational databases. [40] presents approaches ranging from conversion of data in relational databases into a logical form to transferring ILP techniques into other domains for operating on relational databases.

Relational association rule induction, relational decision trees and relational distance based methods are three major approaches of Inductive Logic Programming for relational data mining[41]. Relational association rule induction finds association rules out of data in logical form. WARMR [3] is a well known relational association rule induction algorithm. WARMR is an extended version of propositional Apriori [2] algorithm working on relational data. [42] introduces Mode-Directed Inverse Entailment and Progol [42].

Relational decision tree approach is used for relational classification and regression. SCART [43] and Tilde [44] are well-known decision tree induction algorithms which are extensions of their first order counterparts, CART and C4.5 respectively. The Tilde system implements top-down induction of logical decision trees. SCART algorithm is capable of inducing first-order trees for both classification and regression problems by upgrading the propositional algorithm CART into a relational learner with suitable extensions.

Relational distance based methods adapts propositional statistical methods of classification and regression for relational classification and regression.

2.3.2 Bayesian Networks

Bayesian Networks approach is based on probability theory. The Starting point of this approach is the Bayes theorem which is stated as

$P(H|X) = \frac{P(X|H)*P(H)}{P(X)}$

where P(HIX) is the probability of H to occur given that X occurred and P(H) and P(X) are the probabilities that H and X occur independent of any other conditions.

A Bayesian Network is a directed acyclic graph where each vertex is a variable and edges represent conditional dependencies between the variables[45]. [46, 47] discusses methods for constructing Bayesian net-works from prior knowledge and summarize Bayesian statistical methods for using data to improve Bayesian Network graphical models. [48] discusses general methods of learning Bayesian networks from data and presents connections of Bayesian Networks and Neural Networks.

Bayesian Networks have also been used for multi-relational data mining. [49] proposes an approach to model relational rules using Bayesian Networks. Combining first-order logic and Bayesian Networks is presented in [50]. [51, 52] discuss the problem of learning probabilistic models of relational structure and provides a framework for specifying and learning a probabilistic model of relational structure.

2.3.3 Neural Networks

Neural Networks are the structures which are capable of learning non-linear functions. Building blocks of the Neural Networks are called neurons for their similarity to biological neurons. Neuron has a number of inputs coming from other neurons and an output going to other neurons and each connection between two neurons has a weight. The function of each neuron is to calculate a value using the inputs and their weights, then to produce a result by comparing the calculated value by a threshold value.

Neurons are grouped into layers and these layers are connected for solving several learning tasks including data mining tasks. Using data, weights of connections are updated iteratively and a solution to the problem is found.

Recently, neural networks are proposed for multi-relational data mining. [53] introduces a supervised neural network learning based approach to multi-

relational data mining. Based on a relational database, a neural network is constructed and trained to learn patterns in the data.

2.3.4 Multi-Relational Data Mining

Multi-relational data mining approaches can be categorized according to the kind of data used when mining [54]. For example, ILP requires the data to be in the form of logic clauses. However, most of the data is stored in relational database management systems around the world. Of course, it is possible to generate data in any form (e.g. in the form of logic clauses) from relational databases, but this generation requires extra effort in preprocessing step. Moreover, when format of data enforces a specialized language to be used (e.g. a logic language like Prolog), data mining algorithms implemented in this language will inherently be inefficient. Therefore, there is a need for approaches directly using data stored in relational database management systems for data mining. In this way, efficient languages and efficient built-in database functions such as querying with SQL can be used in data mining.

[3] presents WARMR system that extends single table data mining Apriori algorithm [2] to mine multiple relations for extracting association rules by adapting some techniques from ILP. Discovered association rules are in the form of Prolog queries. This enables the use of variables and other logic notations in Prolog to make queries more expressive.

[1] proposes a framework for multi-relational data mining on relational databases. Relational database's metadata is used to avoid the explosion in the search space providing efficiency and scalability. A graphical language for patterns that can be translated into SQL and other logic languages is described and WARMR is proposed to be solved using this framework as an instance in this work. [55] further improves this framework and presents the efficient discovery of multi-relational decision trees.

Several studies have been performed based on the framework proposed in [1]. [56] uses this framework to implement a Multi-relational Decision Tree Learning

(MRDTL) algorithm for inducing decision trees from relational databases. This algorithm competes with well-known efficient classification algorithms, such as Progol[42], Foil [57, 58] and Tilde [44]. [59] utilizes this framework to use aggregates for relational database propositionalization to apply propositional data mining algorithms. [60] presents an implementation of R-ILA (Relational Inductive Logic Algorithm), which is an adapted version of ILA-2 [61] algorithm for relational data mining. R-ILA algorithm is a covering type of rule discovery algorithm.

CHAPTER 3

MULTI RELATIONAL DATA MINING FRAMEWORK

In this study, a multi-relational data mining implementation for discovering association rules in relational databases is developed based on the framework proposed in [1]. This chapter presents the concepts of this framework.

In multi-relational data mining tasks, the search space explodes as compared to the single relation data mining. Each new table introduced for data mining task increases the complexity depending on the number of attributes it has and the number of distinct values of those attributes. When there is no information about the relations of the attributes, all possible relations between attributes are needed to be considered, which makes the search space huge. For example, there are three relations: Person(Name String, Occupation String, Location String), City(Name String, Mine String, Country String) and Country(Name String, Capital String). When data mining on those relations for association rule extraction (e.g. a rule like "%60 of the people living in a location where there is a coal mine are miners"), there is no information that the Location attribute of the Person relation is a city, but not a country. Indeed, there is no information stating that Occupation attribute is not related to Country relation's Capital attribute. Therefore, all of the possible relations between attributes must be

considered during data mining, and search space becomes huge.

Considering the example given, relations between the Person's Location and City's Name, City's Country and Country's Name, Country's Capital and City's Name are enough to reduce the search space, so that a rule including a statement like "if a person's occupation is equal to capital of the country" will never be considered whereas a rule including a statement like "if a person's location is equal to the city name and there is a mine in this city" is among the rules to be considered.

In relational databases, the relations between the attributes of tables are described by foreign keys. The framework uses foreign keys existing in relational database to reduce the search space when performing data mining tasks. Considering the example given, a foreign key is described on relation Person relating its Location attribute with the Country's Name attribute. Similarly, other foreign key relations are described for other relations.

The framework describes a client-server architecture where client side is responsible for most of the data mining tasks like constructing candidates, selecting among candidates and server side is responsible for efficient processing of requests for a small number of primitives like finding support of a candidate.

3.1 Multiplicity of Associations

The term relation used till now referred to both a table of a relational database and a relation between the records of table. In the remaining of text, discussed subjects are in the context of relational databases. Therefore, a distinction is made, association is used for the relation between the attributes of tables, and table is used as table, that is where records of a type reside.

Associations exist between two tables and have a multiplicity value. Multiplicity describes one-to-many property stating that a record in one table is related to many records in the other table and optional property stating that one record in a table is not required to be related to any record of the other table.

The framework formally defines multiplicity of association A between two tables P and Q with two predicates.

- Multiple(A,P) iff every record in Q may correspond to multiple records in P.

- Zero(A,P) iff a record in Q may have no corresponding record in P.

Multiple predicate states one-to-many property and Zero predicate states the optional property of association. Relational databases allow only associations having certain multiplicity properties. A foreign key association between two tables P,Q, and an attribute in P referencing primary key attribute in Q has the multiplicity property Multiple(A,P), Zero(A,P), not(Multiple(A,Q)), not(Zero(A,Q)). It is impossible to define an association having multiplicity property Multiple(A,P), Multiple(A,Q) with relational databases.

It is possible to extract missing foreign key relations of a database using data to find multiplicity properties from which foreign keys can be extracted. This study assumes that foreign key relations exist in database and can be accessed as meta-data of database. However, multiplicity properties of associations can be used by framework during search phase to improve results.

3.2 Patterns

Data mining on relational databases requires a way of representing a pattern combining multiple tables using associations and conditions of these tables. In single table data mining, the patterns are in the form of a list of "attribute=value" pairs. However with relational database mining, a structural representation of the patterns are required to combine multiple tables with associations and "attribute=value" conditions of tables. Moreover, it should be easy to translate the pattern into a statement of SQL language since the patterns will be run on a database server to evaluate them.

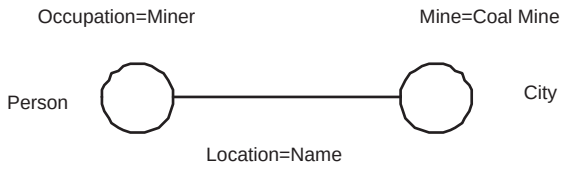

Figure 3.1: Visualization of Selection Graph

The framework proposed in [1] defines Selection Graph as structural representation of patterns used in relational data mining. Selection Graph concept is defined as:

"A selection graph G is a pair (N, E), where N is a set of pairs (t, C), t is a table in the data model and C is a, possibly empty, set of conditions on attributes in t of type t.a operator c; the operator is one of the usual selection operators, =, etc. E is a set of triples (p, q, a) called selection edges, where p and q are selection nodes and a is an association between p.t and q.t in the data model. A selection node n represents a selection of records in the corresponding table n.t which is determined by the set of conditions n.C and the relationship with records in other tables characterized by selection edges connected to n."

Selection nodes are tables in selection graph.Considering the given example with table Person, City and Country, pattern stating "People living in a city where there is a coal mine are miners." has the following corresponding selection graph, which is visualized as in Figure 3.1.

G = (N , E)

N=(*Person*,C1), (City, C2)

C1=(*Occupation = Miner*)

C2=*(Mine=Coal Mine)*

E=*(SN1,SN2, (SN1.Person.Location=SN2.City.Name))*

The selection graphs can be easily translated to SQL or Prolog queries. This makes selection graphs suitable for being used in data mining tasks. During data mining search process, selection graphs are generated as candidates, translated

to SQL query, and this SQL query is passed to the server side to find values of selection graph for the evaluation criteria. The selection graphs that achieve required values for evaluation conditions are further refined to generate other selection graphs and this process continues until no other selection graphs can be generated. To translate a selection graph to an SQL query, a simple algorithm is used[1] (Table 3.1).

Table 3.1: Translating a Selection Graph to SQL Query Algorithm

```
Input:
S //Selection Graph
Output:
SQLString begin table_list = ”
condition_list = ”
join_list = ”
for each node i in selection graph S do
        table_list.add(i.table_name +  T + i)
        for each condition c in i do
                condition_list.add( T + i + . + c)
for each edge e in S do
        join_list.add( e.left_node + . +
        e.left_attribute +  =  +
        e.right_node + . +
        e.right_attribute)
return select distinct  + t0 + . + t0.primary_key +
         from  + table_list +
         where  + join_list +  and  + condition_list
end
```

3.3 Refinements

Selection graphs are generated, evaluated, and then refined in data mining search process. Refining all selection graphs increases the search space of the process. Therefore, only selection graphs having the possibility of yielding ”good” solutions are refined in this study. Determining whether refining a selection graph will yield a good solution is discussed in Chapter 4. There are three ways of

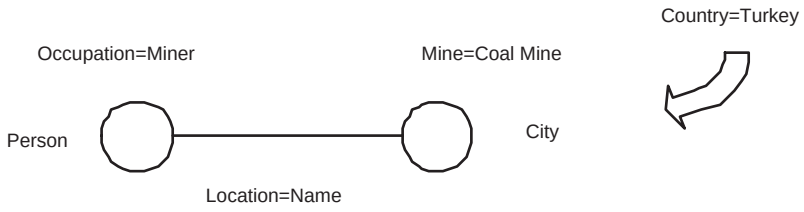

Figure 3.2: Adding a Condition to Selection Graph

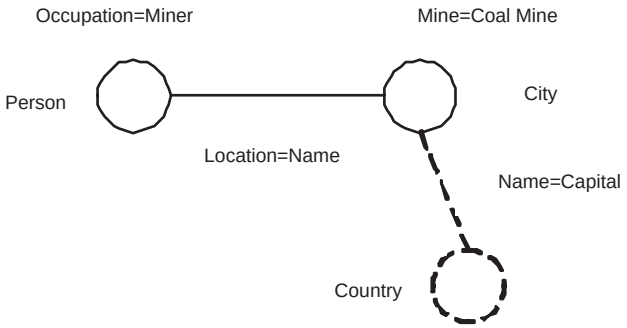

Figure 3.3: Adding an Edge and a Node to Selection Graph

refining a selection graph;

- Adding Condition: This refinement adds a condition to an existing selection node of a selection graph. This refinement does not add any edges to the selection graph. Adding condition "Country=Turkey" to selection graph given in Figure3.2 is an example to this type of refinement.

- Adding Edge and Node: This refinement adds edge and node to a selection graph. This refinement does not add any conditions to the selection graph. Adding "City.Capital=Country.Name" association to the selection edge shown in 3.3 is an example to this kind of refinement. Node Country and selection edge with association City.Country=Country.Name is added to selection graph.

- Add Edge: This refinement adds an edge to a selection graph. No conditions or nodes are added by this refinement. Adding edge with the association "Person.Citizen=Country.Name" is an example to this kind of

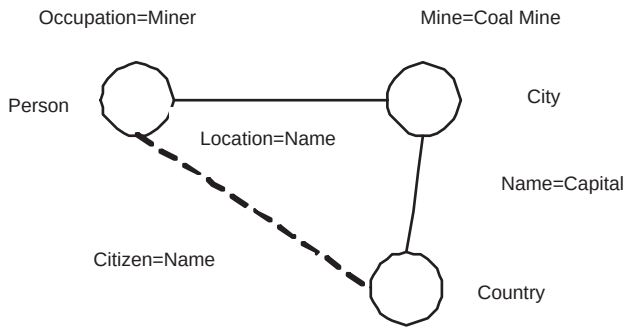

Figure 3.4: Adding an Edge to Selection Graph

refinement. Person and Country nodes already existed, but addition of the association adds no nodes to the selection graph, but only an edge.

3.4 Primitives

This framework defines a set of primitives on the server side to be used during data mining process for acquiring information like number of records satisfying a selection graph from database. The framework allows a selected table to be the *target table* and a selected column of target table to be the *target attribute*. Target table and target attribute are determined by the problem to be solved. For example, when the problem is to find rules describing occupation of a person, Person table is selected as the target table and Occupation attribute is selected as the target attribute. During search process, target table and attribute are used to find the support and confidence values of selection graphs. Support and confidence of a selection graph g with respect to target table t0 and target attribute *attr* is defined as

$$support = \frac{\#\ of\ \ records\ \ in\ \ t0\ \ satisfying\ \ g}{\#\ \ of\ \ \ all\ \ \ records\ \ in\ \ \ t0}$$

$$confidence = \frac{\#\ of\ \ records\ \ in\ \ t0\ \ satisfying\ \ g}{\#\ \ of\ \ \ all\ \ \ records\ \ in\ \ \ t0\ \ satisfying\ \ \ (g{-}attr)}$$

where (g-attr) is the g with condition related to attr is removed. Confidence and support helps to evaluate a selection graph. Primitives helps the calculation of confidence and support of a selection graph.

The framework defines four primitives to be used.

- CountSelection: This primitive is used for calculating the support of selection graph. It returns the result of following query:

  select t0.target, count(distinct t0.primary_key)
  from table_list
  where join_list *and*condition_list *group by t0.target*

  where tablelist, joinlist and conditionlist are calculated as in the Algorithm 3.1 to translate a selection graph to SQL query.

  In this study, CountSelection is the only primitive used. It is used for calculating evaluation criteria of selection graphs (support, confidence).

- MultiRelationalHistogram: This primitive is used to calculate the count of each target attribute value among the records satisfying the selection graph. The framework makes use of this primitive to compute interestingness measures of the selection graph. MultiRelationalHistogram primitive returns the result of following query:

  select t0.target, count(distinct t0.primary_key)
  from table_list
  where join_list and condition_list group by t0.target;

- MultiRelationalCrossTable: This primitive is used to obtain statistical measures for dependency of target attribute and an arbitrary attribute of any table in selection graph. This primitive returns the result of following query:

  select t0.target, ti.cj, count(distinct t0.primary_key)
  from table_list
  where join_list and condition_list group by t0.target, ti.cj;

- MultiRelationalAggregateTable: This primitive is used to compute dependency of target attribute and a numeric attribute of any table in selection graph. It produces the counts and minimum values of numeric attribute

for target attribute values. This primitive returns the result of following query:

select t0.target, m, count(*) from
(select t0.target, t0.primary_key, min(ti.cj) m
from table_list
where join_list and condition_list group by t0.target, t0.primary_key)
group by t0.target, m;

This framework proposes the use of MultiRelationalCrossTable and MultiRelationalAggregateTable primitives for evaluation of Adding Condition type of refinements. However, this task is performed by using CountSelection primitive with simple modifications to selection graph's SQL translation and modifications to SQL query given for this primitive.

CHAPTER 4

IMPLEMENTATION

The implementation of software developed during this study to perform the task of Multi-Relational Data Mining on Relational Databases is presented in this chapter. The implementation is based on the framework proposed in [1], which is discussed in Chapter 3. The framework is mainly used in searching step of the implementation. Starting with an initial selection graph including only the target table as selection node, refinements are generated by adding condition, adding association and edge, or adding edge to initial selection graph. Refinements are evaluated using the CountSelection primitive. Refinements satisfying evaluation criteria form the next level selection nodes to be refined. Process continues until no selection nodes to be refined is left.

Software is implemented in Java programming language using Eclipse Integrated Development Environment (IDE). IBM's DB2 is used as RDBMS. JDBC (Java DataBase Connectivity) solution is used as an interface between Java program and database. JDBC driver used is the one distributed by IBM.

The implementation consists of four steps; preprocessing, running Apriori algorithm [2] on tables, searching and rule generation. In the preprocessing step, the software inputs the database's meta-data and asks user for target table, target attribute and attribute properties of the tables. Then, a graph displaying the associations between tables is displayed. The user can delete

the tables that are irrelevant. Once the relevant tables are identified in the first step, Apriori algorithm is run for each of these tables in second step. The Apriori algorithm is modified such that support is calculated on target table using associations existing between the table on which Apriori runs and target table. Running Apriori algorithm on tables yields a number of large item sets for each table. Large item sets found are used to refine selection graphs in the third step. Selection graphs are generated, evaluated and the refinements of selection graphs, which are also selection graphs are generated. The search continues until no selection graph is left. At this step, a number of "good" candidates with high possibility of generating "good" rules are found. Finally, "good" rules are extracted from "good" candidates found in third step. Being a "good" rule and "good" candidate is explained later in this chapter.

4.1 Preprocessing

In this step, the software inputs the meta-data of relational database on which data mining operation will be performed, lets the user to select the relevant tables, relevant attributes and types of the attributes. Then distinct values of the categorical attributes are obtained from database and values of continuous attributes are discretized.

4.1.1 Reading Meta Data of Database

When software is started, it reads the meta-data of the database using JDBC API. Name of tables, attributes of tables, database types of attributes (e.g. BOOLEAN, VARCHAR, INTEGER etc.), primary key and foreign key constraints of tables are the information retrieved from meta data of database. Then, a screen shown in Figure5.1 is opened presenting meta data information to user. User selects one of tables as target table and one of attributes of target table as target attribute. Selection of target attribute is optional depending on whether a target attribute is required by the problem. For example, in a classification problem, target attribute is selected as the attribute on which

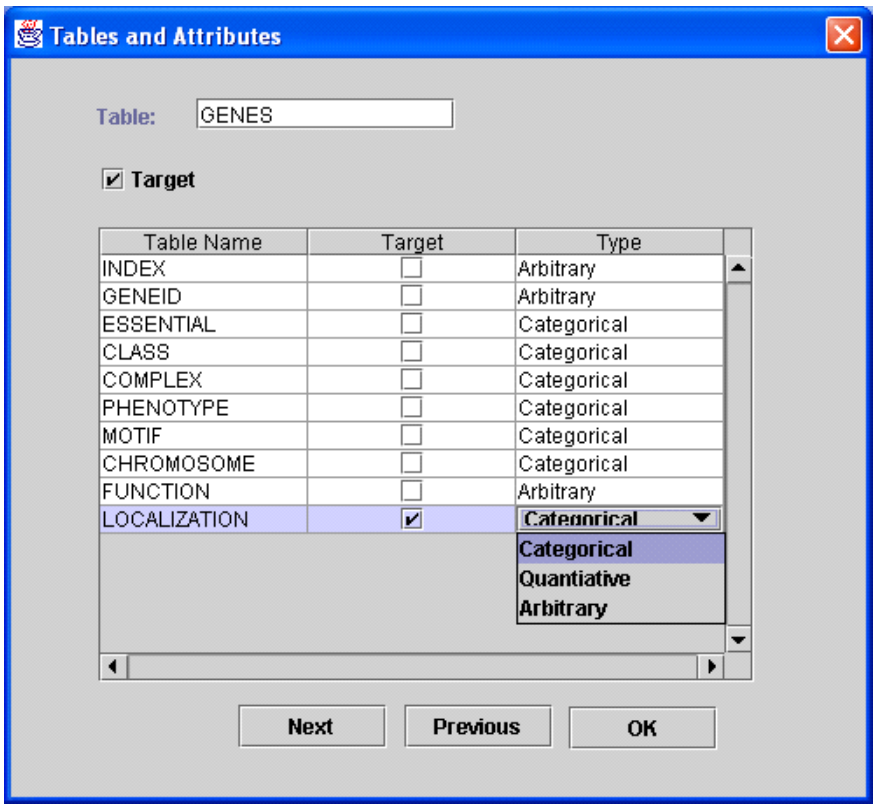

Figure 4.1: User Interface for Target Table, Target Attribute and Attribute Type Selection

classification is performed.

Afterwards, user selects the type of attributes depending on whether they are categorical, numeric or arbitrary. Type of an attribute which is irrelevant in data mining task is selected as arbitrary meaning that it will be ignored in data mining operation unless it is a part of a primary key or a foreign key constraint. An example to such an attribute would be photograph of a person stored in personnel table of a company. Type of an attribute is selected as categorical if attribute stores a distinct categorical information and this information is useful in data mining operation. For example, position attribute of a person stored in personnel table may have distinct categorical values like "Software Engineer", "System Engineer", etc. Therefore, type of this attribute is selected as Categorical.

It is possible that a categorical attribute's database type be INTEGER if

this attribute stores categorical information like the grade of a student from a course which can take the values {0,1,2,3,4,5}. Attribute's type is selected as numerical if this attribute stores a numeric value. Treating it as a categorical attribute is not feasible such that it is required to handle interval values of this attribute. Salary attribute of a Personnel table of a company is an example to numerical type of attribute.

4.1.2 Analyzing Associations and Resolving Recursiveness

Once meta-data of database is read, then associations between tables are analyzed and a graph with recursive associations resolved is constructed.

During analysis, aliases are used for tables. This allows a table to appear multiple times in the graph, which enables resolution of recursiveness between tables. The following two examples describe two databases, one including recursive associations and one without any recursive associations. They are used for describing analysis and construction phases in this section.

Database DB_1 with recursive associations, target table A.

Table A with attributes A_1,A_2 : A(A_1,A_2)

Table B with attributes B_1,B_2 : B(B_1,B_2)

Primary keys: A.A_1

Foreign keys: B.B_1 refers to A.A_1, B.B_2 refers to A.A_1

Database DB_2 without recursive associations, target table C

Table C with attributes C_1 : C(C_1)

Table D with attributes D_1,D_2 : D(D_1,D_2)

Table E with attributes E_1,E_2,E_3 : E(E_1,E_2,E_3)

Primary Keys : C.C_1, D.D_1, E.E_1

Foreign Keys : D.D_2 refers to C.C_1, E.E_2 refers to C.C_1, E.E_3 refers to D.D_1

Assuming aliases are not used to resolve recursiveness for the moment, graph construction starts with the target table. Target table is added as a node to the graph. Associations of target table with other tables are found. A node is added for each of those associating tables and associations are added as edges between

target table and other table involved in association. A breadth-first construction is followed such that associations closer to the target table are added first.

Considering DB_1, construction is performed starting with target table A. Association A.A_1 = B.B_1 (first foreign key constraint) is added to the graph first. Therefore, B table is also included in the graph. Then, looking at B table's associations, B.B_2=A.A_1 is added. The associations included are A.A_1=B.B_1 and B.B_2=A.A_1 which implies B.B_1=B.B_2. However this implication is meaningless, which makes DB_1 recursive.

Considering DB_2, analysis is performed starting with the target table C. First C.C_1=D.D_2 is added to the graph, D table is included in the graph. Then C.C_1 = E.E_2 is added and E is included in graph. Adding last foreign key constraint E.E_3=D.D_1 of already added tables D,E, all associations in the graph are C.C_1=D.D_2, C.C_1=E.E_2 and E.E_3=D.D_1. Since there are no meaningless implications in those associations, DB_2 is not recursive.

To solve recursiveness problem, aliases are used for tables during construction. Recursiveness problem arises when addition of an edge implies two different attributes of the same table to be equal to each other. In this case, an alias is created for this table, and an association is obtained with the new alias. Considering DB_1, recursiveness problem is solved using aliases as described. Target table A is added to the graph with alias t0. Association $t_o.A_1$=$t_1.B_1$ is added to the graph, where t_1 is table B added to graph with this association. Then, $t_1.B_2$=$t_0.A_1$ is added, which yields a recursiveness problem. The association causing the problem, which is $t_1.B_2$=$t_0.A_1$, is removed and t_2 is added as an alias for table A. Then association is added as $t_1.B_2$=$t_2.A_1$. The algorithm for graph construction with aliases is described in Table4.1.

When the graph is constructed, it is displayed to user using JGraphT. User then analyzes the graph, and deletes tables that are irrelevant in data mining operation.

Table 4.1: Graph Construction Algorithm

```
Input:
targetTable Output:
graph
begin
graph = empty addNodeToGraph(targetTable, t0);
while true
        associations=findAssociationsToAddBreadthFirst();
        if associations is empty
                return
        for each association
                conflicts = addEdgeToGraph(association)
                if conflicts
                        removeAssociation(association)
                        createNewAlias()
                        modifyAssociationForNewAlias()
                        addEdgeToGraph(association)
return graph
end
```

4.1.3 Discretization of Continuous Attributes

The user chooses types of continuous attributes as Numerical so that they are discretized, intervals are formed and those intervals are used to add conditions on continuous attributes to selection graphs in searching step.

Several methods have been used for discretization of continuous attributes. Equal-width interval binning, equal-depth interval binning and recursive minimal entropy partitioning are the most-widely known methods for this task.

Equal-width partitioning method is the simplest method for discretization. Using a user supplied parameter for number of bins, NB, interval width of bins is found as binWidth = (maximumValue - minimumValue) / NB. Intervals are formed having binWidth size starting from the minimum value and ending with maximum value.

Equal-depth partitioning method orders the samples according to the continuous attribute to be discretized. The user supplied parameter for number

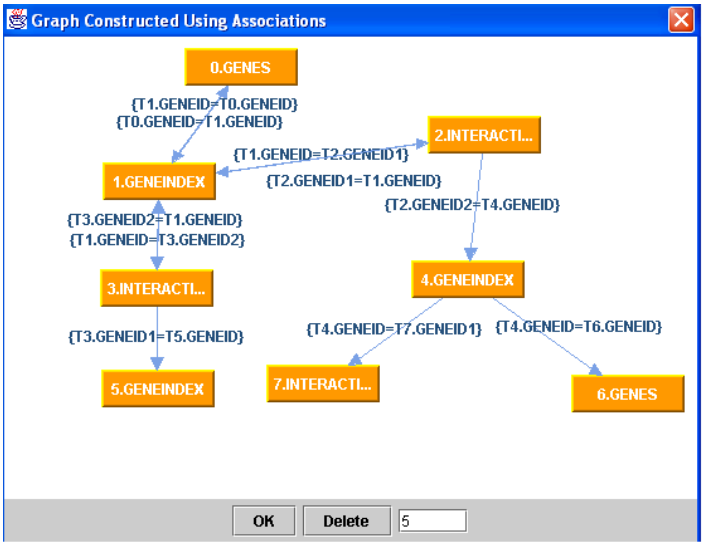

Figure 4.2: Graph Constructed Using Associations Between Tables

of bins, NB, is used to calculate the number of samples to be placed into each bin, called depth of bin as binDepth = numberOfSamples / NB. Then bins are formed by placing samples into intervals starting with the minimum valued sample and ending with the maximum valued sample such that each interval has binDepth number of samples. Interval bounds are the values of minimum valued and maximum valued samples of each bin.

Recursive minimal entropy partitioning method is proposed in [62]. This algorithm uses class entropy of candidate partitions for finding interval boundaries. To find the intervals of a continuous attribute a with values from v_1 to v_n, a bin boundary v_k is found such that the class information entropy is minimized for intervals (v_1,v_k),(v_k,v_n) of this attribute. Then the same technique is recursively applied to v_1 and v_k, v_k and v_n, and so on until a termination criterion is met. A Minimum Description Length (MDL) principle is defined as the termination criterion for entropy partitioning method.

Considering three methods, Equal-width partitioning is a simple unsupervised method and recursive minimal entropy is a supervised method[63]. Equal-depth partitioning is also a supervised method. Equal-width partitioning is negatively affected by outliers. For example considering samples with values

2,3,4,5,6,100 for a continuous attribute and assuming ten bins, first 5 samples are placed into the first bin, last sample is placed into the last bin, and 8 bins are left empty. However, using equal depth partitioning, samples are scaled well. [64] proves that given any specified number of partitions, equal depth partitioning method minimizes the partial completeness level, which is a measure of information lost by partitioning. Moreover, equal-depth partitioning method is simpler than recursive minimal entropy partitioning method. Equal depth partitioning is used to discretize continuous attributes in this study.

In real world, it is very common that some of the attributes of a record have missing values. In fact, all the attributes of a record can be missing if this attribute is not specified to be "not null" in database description. Dealing with missing attributes in a learning task is a common problem for all learning algorithms. [65] describes the arising problems due to missing values of attributes. These problems can be categorized into two categories considering the approach in this study.

- How attributes of missing values are compared in training set? Will two records having a null value for attribute A be treated as having the same value for attribute A?

- When testing a record, how are missing values of this record will be treated?

One solution for the first problem is ignoring cases in training set with missing value of attribute A when finding a rule containing attribute A. Another solution is to fill in missing values with A attribute's most common value. There is another solution by filling missing values of A with a value calculated using the values of other attributes. Solutions for second problem depends on the solution selected for the first problem. If cases for missing values are ignored in training data, this attribute's value is also ignored in testing. A test case with a missing value for attribute A can not be described by a rule containing a condition of attribute A. Similarly, missing attributes of a test case can be filled with most

common value or a value calculated using other attributes. Solutions by filling missing attributes perform better than the solutions ignoring the missing values.

In this study, missing attribute values are not filled, neither they are ignored. Missing attribute values are treated to be same. Therefore, rules containing condition Attribute.value=null are discovered and test cases are evaluated against those rules as well.

4.2 Pruning Search Space Utilizing Apriori Algorithm

Once tables, attributes, and attribute types are selected and graph is constructed in previous step, values for the selected attributes are read for categorical attributes. Numeric values are discretized and Apriori algorithm [2]is run for each selected table in this step.

In searching step, selection graphs are refined by *adding a condition*, *adding an edge and a node*, or *adding an edge*. Only *adding a condition* refinement adds conditions of the form *attribute=value* to selection graph. When refining is done by adding a condition, all attributes of the tables included in selection graph must be considered. If a condition for an attribute *attr* doesn't exist in selection graph, selection graph is refined to have *attr=value* for all values of this attribute. However, when support of selection graph with the only condition *attr=aValue* is smaller than the required support, adding this condition to any selection graph in a refinement will result in selection graph with support smaller than the required value. As a result, when refining, adding this condition to any selection graph increases search space. In general, when a selection graph has a subset of its conditions that can not achieve required support value, this selection graph also can not achieve required support value neither. This property is described by Apriori algorithm. Apriori algorithm efficiently produces sets of conditions for a single table satisfying required support value. Therefore, Apriori algorithm is used to find conditions that are added when refining a selection graph for each table.

Apriori algorithm presented in [2] generates all sets of conditions having

support greater than a certain support value, called *minsupport*. Sets of conditions having support greater than minsupport are called large itemsets and ones having a smaller support than minsupport are called small itemsets. Apriori algorithm generates all large itemsets efficiently.

Apriori algorithm presented in [66] is implemented in this study with modifications. To find the support of an itemset when running Apriori on a table, conditions in itemset and associations between this table and the target table form a selection graph and CountSelection primitive is called with SQL query of selection graph. As a result, support of an itemset of a table t_i is calculated by counting the records of target table t_0 that are related to records of t1 satisfying the conditions of itemset. Support is calculated against the target table, since, in searching step, support of every selection graph is calculated against to target table. An itemset for table t_i found to be large with support calculated against t_i may possibly be a small itemset with support calculated against target table t_0. Since the implementation discovers association rules related to target table, support of itemsets are calculated against the target table.

When calculating support of itemsets of table t_i against target table t_0, the records of t_0 related to records of t_i satisfying conditions in itemset are found using the associations existing in distinct paths from t_i to t_0. Distinct paths from t_i to t_0 are found by applying an algorithm on graph constructed in preprocessing step. This algorithm finds all paths from t_i to t_0 having no repeating nodes.

Table 4.2: Apriori-Gen Algorithm

```
Input:
Li-1 //Large itemsets of size i-1
Output:
Ci //Candidates of size i
begin
Ci = {}
for each I of Li-1 do
        for each J of Li-1 different than I
                if i-2 of elements in I and J are equal then
                        Ck = Ck U {I U J}
```

Apriori algorithm presented in [66] is modified for support calculation and the

Table 4.3: Apriori Algorithm

```
Input:
I //Initial Itemsets consisting of one condition
D //Database of transactions
s // minsupport
ti //alias of table
Output:
L //Large Itemsets
begin
k = 0; //k is used as the scan number
L={}
C1=I //Initial candidates are set to be the initial itemsets
distinctPaths=findDistinctPathFor(ti) //distinct paths found
repeat
      k=k+1;
      Lk={}
      for each Ii of Ck do
            selectionGraphs=formSelectionGraph(distinctPaths, Ii)
            for each selectionGraph of selectionGraphs
                  support=CountSelection(selectionGraph)
                  if support > minsupport then
                        Lk=Lk U Ii
      Ck+1=Apriori-Gen(Lk)
until Ck+1 = {}
end
```

resulting algorithm is presented in Tables 4.2 and 4.3. This algorithm is run for all table aliases selected to be involved in data mining operation. Large itemsets of all aliases are stored in memory to be used during searching step.

4.3 Searching

Searching in search space of selection graphs is performed, results are evaluated, and selection graphs satisfying evaluation criteria are found in this step.

Initially, searching starts by forming an initial selection graph with one node for the target table, which is the only selection graph in level 0. Then refine-

ments of this selection graph for level 1 are found by adding large itemsets of target table found in previous step using Apriori algorithm by *adding condition* refinement. To find refinements in level 2, selection graphs are evaluated against criteria which is described later and satisfactory selection graphs are found. Then, next closest node to the selection graph is found, which is the one with minimum alias not existing in the selection graph. Associations of this node added to the selection graph using *adding node and edge* condition. If there exist two nodes in selection graph having associations between them and those associations are not added to selection graph yet, they are added to the selection graph using *adding edge* refinement. Large itemsets of the closest node are added to selection graph to form refined selection graphs using *adding condition* refinement to form level 2 selection graphs. Searching process continues until there is no selection graph to refine. The search algorithm described is presented in Table4.4. Satisfying refinements are found by evaluating refinements against

Table 4.4: Searching Algorithm

```
Input:
No Input
Output:
result //Selection Graphs satisfying evaluation criteria begin
result
selectionGraph = formInitialSelectionGraph();
refinements = findRefinementsAddingLargeItemSets(selectionGraph)
satisfyingRefinements = evaluateRefinements(refinements);
result.add(satisfyingRefinements);
while satisfyingRefinements is not empty
      closestNode = findClosestNodeTo(satisfyingRefinements);
      refinements = addAssociationsForClosestNode(satisfyingRefinements);
      addLargeItemSetsOfClosestTo(refinements);
      satisfyingRefinements = evaluateRefinements(refinements);
      result.add(satisfyingRefinements);
return result;
end
```

criteria depending on the problem to be solved. If there is no target attribute

specified, refinements are sorted according to the support of refinements found by using CountSelection primitive. Then a user provided number of refinements having high support values are selected as satisfying refinements and passed to next level for refinement and evaluation. If a target attribute is specified by the user, then refinements are sorted according to their "fmeasure" values and a user supplied number of refinements are passed to the next level. "fmeasure" is a method to evaluate rules in Weka [67] with respect to a particular class. "fmeasure" value is calculated by using following formulas.

fmeasure = (2 * recall * precision) / (recall + precision)

recall = (correctly classified positives) / (total positives)

precision = (correctly classified positives) / (total predicted as positive)

When searching is finished, good selection graphs are determined. If a target attribute is specified, If-Then rules are extracted out of good selection graphs easily. Condition of the target forms If part and all other conditions and associations of selection graph form Then part. These rules are evaluated against a user specified confidence value using CountSelection primitive. Rules satisfying this confidence value are chosen to be the good rules.

When no target attribute is specified, rules are extracted from refinements as described in [68]. To extract rules from a large itemset l, all non-empty subsets of l are found. Then for each such subset a, support of l and a are computed, if (support of l) / (support of a) which is the confidence of rule If a then l - a is greater than a user supplied confidence, then a good rule is found. In this study, rules are extracted from selection graphs in a similar way. All non-empty subsets of a selection graph are found such that for each subset a, all conditions of l - a are the conditions of attributes of the target table. Then support values for l and a are calculated using CountSelection primitive. Good rules satisfying the specified confidence value are found.

A target attribute is specified when data mining problem is a classification or regression problem. The implementation developed in this study discovers association rules hidden in relational databases. The implementation is not specified

for classification task. Therefore, when a classification problem is given, association rules are discovered for the classification attribute. When classifying a new instance, discovered association rules are searched to find the one with the highest confidence value. However, there are some specialized methods for classification problems as in decision trees, which are not used in this study. Bagging predictors is such a method proposed and used in classification and regression trees[69]. This method generates multiple versions of predictors and use these versions to form an aggregated predictor. The aggregation averages over the versions for prediction of continuous attributes and does a majority voting for prediction of categorical attributes, which are used in regression and classification respectively. [70] provides two explanations for error rate reduction using bagging both based on Bayesian learning theory. [71] reports results of applying Bagging predictors method on C4.5 decision tree learning algorithm. An empirical comparison of algorithms using voting for classification problems is presented in [72]. However none of the majority voting algorithms are used in this study. Therefore, for a classification problem, best association rule (predictor) is used to find the classification of an instance. As a result, the implementation shows worse performance on classification problems compared to well known classification methods.

CHAPTER 5

TEST RESULTS

This chapter presents test results using the localization prediction task of KDD Cup 2001[4]. A training dataset is provided on which software runs to find good rules for localization attribute of genes, and a test set is provided on which good rules are applied to predict the localization of genes.

Database design for given the datasets is presented in Figure5.1. GeneIndex table and index attribute of genes table don't exist in the original genes dataset provided. They are added to normalize the database to make genes table have a primary key. Genes table stores a variety of details , e.g. chromosome, phenotype, motif, etc., about the various genes of one particular type of organism. Interactions table stores the information of interacting genes, interaction types, and correlation value of interaction. Tests are performed on a Intel(R) Pentium(R) 4 CPU 1.60 GHz PC with 512 MB of RAM. Training dataset and test dataset is stored on IBM DB2 database in seperate databases, TRAIN and TEST. During tests, parameter values in Table 5.1. Size of the search space depends on the number distinct values of the attributes of the training dataset and tables involved in data mining. Number of distinct values of the attributes for training dataset is shown in Table 5.2. The task is to find rules describing the categorical Localization attribute having 15 distinct values. Distribution of samples to 15 distinct values for target attribute Localization is {(cell wall,3),

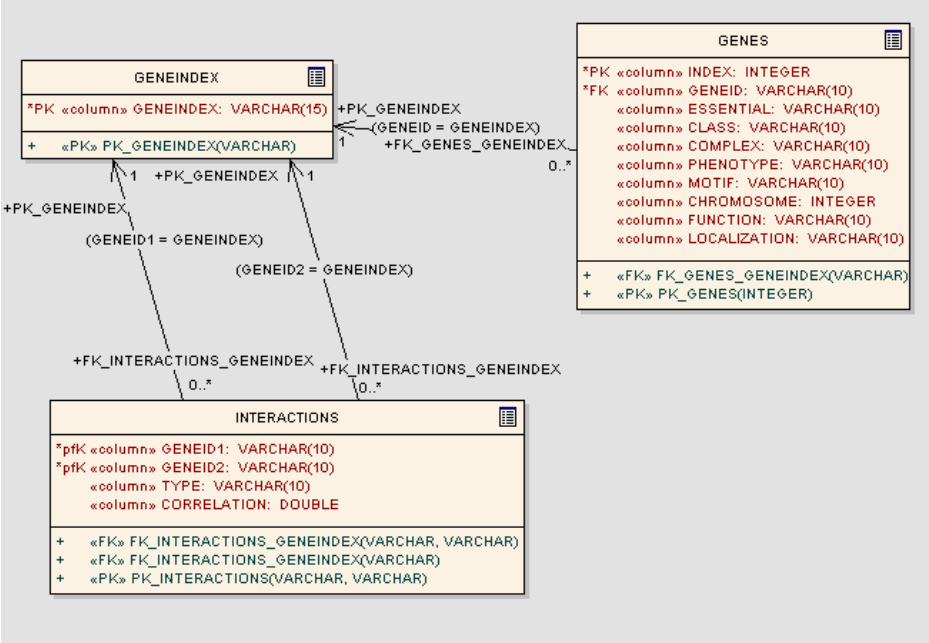

Figure 5.1: Database Design of GENES and TEST Databases

Table 5.1: Values of Parameters Used During Test

Name	Value	Description
Support	0.0002	Support calculated against target table
FMeasure	0.05	FMeasure calculated for a class
Expand Size	50	Number of rules to expand in each level
Table Conditions	3	Max. number of conditions for each table
All Conditions	4	Max. number of conditions for a rule
Number of Bins	20	Number of bins for a continuous attribute

(cytoplasm,931), (cytoskeleton,389), (endosome,23), (extracellular,11),(er, 239), (golgi,175), (integral membrane,4), (lipid particles,2), (mithocondria, 346), (nucleus, 1671), (peroxisome,37), (plasma membrane, 328), (transport vesicles, 52), (vocuole, 135)}. Training dataset and test dataset have many records containing missing values for attributes. These missing values are treated as having the same value, which is "null". As a result, rules containing attribute="null" are discovered during data mining process.

Table 5.2: Distinct Values for Attributes of Tables in Dataset

Attribute	Number of Distinct Values	Selected Type
GeneIndex.GeneId	862	Arbitrary
Genes.Index	4346	Arbitrary
Genes.GeneId	862	Arbitrary
Genes.Essential	4	Categorical
GENES.Complex	52	Categorical
Genes.Phenotype	13	Categorical
Genes.Motif	236	Categorical
Genes.Chromosome	17	Categorical
Genes.Function	13	Arbitrary
Genes.Localization	15	Categorical
Interactions.GeneId1	460	Arbitrary
Interactions.GeneId2	453	Arbitrary
Interactions.Type	3	Categorical
Interactions.Correlation	817 values between -1,+1	Numerical

5.1 Tests Without Using Relational Information

Tests are performed on training dataset without using relational information of the given dataset. The results of this set are the same as if Apriori algorithm is run on single table, GENES (Table5.3). Tests are performed on test

Table 5.3: Results Without Using Relational Information on Training Dataset

Confidence	Time (Seconds)	Rules Found	All	Correct	Incorrect	Not Found	Ratio
0.5	1030	3159	4347	3735	490	121	%85.9
0.7	1030	2901	4347	3675	207	464	%84.5
0.9	1030	2733	4347	3602	86	658	%82.9

dataset without using relational information of given dataset. Therefore, results of this set are the same as if Apriori algorithm is run on single table, GENES. Results of this set of tests are shown in Table5.4. Best accuracy ratio obtained without using relational information on training dataset is %85.9 and

Table 5.4: Results Without Using Relational Information on Test Dataset

Confidence	Time (Seconds)	Rules Found	All	Correct	Incorrect	Not Found	Ratio
0.5	1030	3159	383	226	124	32	%59.0
0.7	1030	2901	383	209	83	90	%54.5
0.9	1030	2733	383	197	74	111	%51.4

this ratio is achieved using a confidence value 0.5. When confidence value is increased, number of unpredicted cases increases whereas number of incorrect prediction decreases. Using the confidence value achieving best accuracy on training dataset, classification task is performed in test dataset achieving an accuracy of %59.0. Considering the accuracy values in test dataset, this is the best ratio to be obtained.

5.2 Tests Using Relational Information

Tests are performed on training dataset using relational information of dataset. Tables GENEINDEX, GENES and INTERACTIONS are used for rule discovery for target attribute Localization. Results of this set of tests are shown in Table5.5.

Best ratio is achieved with 0.5 confidence value. In general, when confidence increases, the number of incorrect predictions decreases and the number of cases with no prediction increases. This behavior is expected since when confidence is low, then the number of rules for prediction increases. Hence, more rules are used to predict the "Localization" of a sample increasing the rate of total predictions as well as incorrect predictions. Tests are performed on the test dataset using relational information of dataset. Tables GENEINDEX, GENES, and INTERACTIONS are used for rule discovery for target attribute "Localization". Results of this set of tests are shown in Table5.6. Best accuracy ratio obtained using relational information on training dataset is %86.3 and this ratio

Table 5.5: Results Using Relational Information on Training Dataset

Confidence	Time (Seconds)	Rules Found	All	Correct	Incorrect	Not Found	Ratio
0.5	1232	3581	4347	3754	508	84	%86.3
0.7	1232	3187	4347	3738	213	395	%86.0
0.9	1232	2977	4347	3639	74	633	%83.7

Table 5.6: Results Using Relational Information on Test Dataset

Confidence	Time (Seconds)	Rules Found	All	Correct	Incorrect	Not Found	Ratio
0.5	1232	3581	383	228	136	18	%59.5
0.7	1232	3187	383	212	99	71	%55.3
0.9	1232	2977	383	199	80	103	%51.9

is achieved using a confidence value 0.5. When confidence value is increased, number of unpredicted cases tends to increase and the number of incorrect prediction tends to decrease as in single table case. Using the confidence value achieving best accuracy on training dataset, classification task is performed in test dataset achieving an accuracy of %59.5, which is better than the single table case. Considering the accuracy values in test dataset, this is the best ratio to be obtained.

5.3 Winner's Approach

Approach, implementation and test results of the winner for Localization task of KDD Cup 2001 are presented in this section. Information about the winner presented in this section is based on [73].

The training data includes many missing values for the attributes. Missing attributes are compensated by using the information of three attributes of GENES table (Class, Complex, Motif) and INTERACTIONS table since those attributes and relation are strongly correlated the Localization attribute.

In data analysis, three different approaches are applied. First approach discovers association rules strongly relevant to the attribute Localization. Second approach makes combinations of discovered association rules to form a predictor using boosting method. Third approach uses nearest neighbor method. Third approach results in better results than other two in training dataset. Therefore, nearest neighbour method is chosen for finding Localization values of test dataset.

Then, using a permutation of four attributes, tests are performed to find optimal set of four attributes yielding highest performance on training dataset. The optimal set of attributes are found to be {Complex, Class, Interaction, Motif} for the predicting Localization attribute. Then the predictors found by applying nearest neighbor method with optimal four attributes to training dataset are used to predict values of cases in test dataset.

Testing achieves an accuracy of %72.2, which is better than the results of implementation developed in this study. This is an expected result since using association rule discovery algorithms for a classification problem is not a good choice. Also, association rules found in this study are not combined to generate good predictors using methods such as bagging or boosting, which may increase the classification ability of the implementation.

5.4 Discussion

Test results have shown that the accuracy is increased when relational information is used. Results obtained without using relational information are the same as the results that could be obtained running Apriori algorithm on single table. This is a property of the implementation and the framework proposed in [1]. This framework supports a range of multi-relational data mining algorithms which are direct generalization of their propositional (attribute-value) counterparts. For this reason, the implementation developed in this study is a generalization of Apriori algorithm and when the implementation is run on single table, the result is the same as running Apriori algorithm on single table.

Comparing the results of the implementation with the results of winner approach for Localization task of KDD Cup 2001, it is observed that winner approach achieves %12.7 higher accuracy rate. This is an expected behavior since the implementation in this study performs the task of association rule discovery. And discovered rules are used for classification of Localization attribute by selecting the association rule consistent with test case having highest confidence value as predictor. However, winner applies algorithms which are specialized in classification problems.

In overall, the implementation in this study performs well considering that the problem to be solved is a classification problem and the implementation in this study is an association rule discovery algorithm having no specialized features for the classification problems.

CHAPTER 6

CONCLUSION

A relational data mining software operating on relational databases is developed in this study. The software is based on a multi-relational data mining framework proposed in [1] and discovers association rules related to target table from a relational database. The framework proposes using meta-data (i.e. primary and foreign key constraints) for pruning search space.

In order to find rules in a relational database, meta-data of the database is read first. The user selects the target table, target attributes and types of the attributes. Associations between tables in relational database are analyzed, recursive cases are resolved, and a graph is constructed, which is presented to the user for fine tuning. The framework excludes discovery of recursive descriptions for the patterns. This study improves the framework by solving recursive associations and generating recursive descriptions for the patterns.

Apriori algorithm is implemented to find large itemsets of each table. Large itemsets are used for refining selection graphs. Before Apriori algorithm is applied, distinct values of columns are read from database and continuous attributes are discretized to form intervals using equal-depth partitioning method. Support calculation in Apriori algorithm is modified so that the support is calculated against the target table. To calculate support against target table, an algorithm is developed to find all distinct paths from a table to target table.

In searching step, selection graphs are refined using three refinements proposed by framework. Refinements are evaluated using "fmeasure" method, which provides a measure to select selection graphs to refine. A specified number of selection graphs are refined in each level of searching to further prune the search space. Searching stops when there exists no selection graphs to refine. Then, rules are generated from the selection graphs satisfying "fmeasure" and confidence criteria.

Developed software is tested with KDD Cup 2001's Localization task and results are presented. Developed software yielded %59 accuracy rate in best case without using relational information in training dataset. Using relational information in training dataset, best accuracy rate is found to be %59.5. These results show that using relational information achieves higher accuracy rates.

Comparing the results of the implementation with the results of the winner approach, winner approach achieves an accuracy rate %12.7 higher than the accuracy rate achieved in this study. This is an expected result since the winner approach uses specialized algorithms for classification task. In overall, the implementation developed in this study performs well considering that the purpose of the implementation is to discover association rules in relational databases, not classification.

There are some limitations of this study which are inherited from the framework used. Rules involving *Absence* and *Aggregate* concepts can not be discovered using the implementation in this study. In absence case, patterns including conditions on absence of related records can not be discovered. For example, considering two relations Personnel and Department, the rule having condition "personnel not working in Quality Assurance Department" can not be discovered. In aggregate case, patterns including conditions over groups of related records can not be discovered. For example, a rule having condition "a personnel working in three different departments" can not be discovered either using the implementation developed in this study. As a future work, these limitation can be eliminated.

Association rules discovered using the implementation developed in this study are not suitable for classification tasks. Methods like Bagging proposed in [69] can be used to generate better predictors using association rules. Generating classifiers from association rules can also be a future work of this study.

Missing values of attributes are treated as having the same value in this study. However, [65] states that filling missing values of an attribute with the common value of the attribute or calculating the missing value of the attribute using values of other attributes achieves better accuracy rates. Using a method for filling missing values can be a part of a future work of this study.

REFERENCES

[1] Arno J. Knobbe, Hendrik Blockeel, Arno P. J. M. Siebes, and D. M. G. van der Wallen. Multi-relational Data Mining. Technical Report INS-R9908, 31, 1999.

[2] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining Association Rules between Sets of Items in Large Databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., 26–28 1993.

[3] L. Dehaspe and L. De Raedt. Mining Association Rules in Multiple Relations. In S. Džeroski and N. Lavrač, editors, *Proceedings of the 7th International Workshop on Inductive Logic Programming*, volume 1297, pages 125–132. Springer-Verlag, 1997.

[4] University of Wisconsin CS Department. KDD Cup 2001, http://www.cs.wisc.edu/ dpage/kddcup2001/, Last Changed on September 19 2001, Last Accessed on August 20 2004.

[5] W. J. Frawley, G. Piatetsky-Shapiro, and C. J. Matheus. Knowledge Discovery in Databases - An Overview. *Ai Magazine*, 13:57–70, 1992.

[6] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, 17:37–54, 1996.

[7] Smyth Padhraic Fayyad Usama, Piatetsky-Shapiro Gregory. The KDD Process for Extracting Useful Knowledge from Volumes of Data. In *Communication of the ACM*, volume 29, pages 27–34, November 1996.

[8] Michael Goebel and Le Gruenwald. A Survey of Data Mining and Knowledge Discovery Software Tools. *SIGKDD Explorations*, 1(1):20–33, 1999.

[9] Charles X. Ling and Chenghui Li. Data Mining for Direct Marketing: Problems and Solutions. In *Knowledge Discovery and Data Mining*, pages 73–79, 1998.

[10] R. Potharst, U. Kaymak, and W. H. L. M. Pijls. Neural Networks for Target Selection in Direct Marketing. 2001.

[11] M. Richardson and P. Domingos. Mining Knowledge-Sharing Sites for Viral Marketing, 2002.

[12] Peter van der Putten. Data Mining in Direct Marketing Databases. In W. Baets, editor, *Complexity and Management: A Collection of Essays*. World Scientific Publishers, 1999.

[13] Wenke Lee and Salvatore Stolfo. Data Mining Approaches for Intrusion Detection. In *Proceedings of the 7th USENIX Security Symposium*, San Antonio, TX, 1998.

[14] Eric Bloedorn, Alan D. Christiansen, Willian Hill, Clement Skorupka, Lisa M. Talbot, and Jonathan Tivel. Data Mining for Network Intrusion Detection: How to Get Started, August 2001.

[15] Wenke Lee, Salvatore J. Stolfo, and Kui W. Mok. A Data Mining Framework for Building Intrusion Detection Models. In *IEEE Symposium on Security and Privacy*, pages 120–132, 1999.

[16] Wenke Lee, Salvatore J. Stolfo, and Philip K. Chan. Learning Patterns from Unix Process Execution Traces for Intrusion Detection. In *Proceedings of the AAAI97 workshop on AI Approaches to Fraud Detection and Risk Management*, pages 50–56. AAAI Press, 1997.

[17] D. Malerba and F. Lisi. An ILP Method for Spatial Association Rule Mining, 2001.

[18] Vladimir Estivill-Castro and Michael E. Houle. Robust Distance-Based Clustering with Applications to Spatial Data Mining. *Algorithmica*, 30(2):216–242, 2001.

[19] D. Malerba, F. Esposito, F. Lisi, and A. Appice. Mining Spatial Association Rules in Census Data, 2002.

[20] Donato Malerba and Francesca A. Lisi. Discovering Associations between Spatial Objects: An ILP Application. *Lecture Notes in Computer Science*, 2157:156–??, 2001.

[21] Martin Ester, Hans-Peter Kriegel, and Jorg Sander. Spatial Data Mining: A Database Approach. In Michel Scholl and Agns Voisard, editors, *Fifth Symposium on Large Spatial Databases (SSD'97)*, volume 1262, pages 48–66, Berlin, Germany, 1997. Springer.

[22] W. Wang, J. Yang, and R Muntz. STING+: An Approach to Active Spatial Data Mining. In *Fifteenth International Conference on Data Engineering*, pages 116–125, Sydney, Australia, 1999. IEEE Computer Society.

[23] Shi Wang, Wen Gao, Jintao Li, Tiejun Huang, and Hui Xie. Web Clustering and Association Rule Discovery for Web Broadcast. In *Web-Age Information Management*, pages 227–232, 2000.

[24] Kosala and Blockeel. Web Mining Research: A Survey. *SIGKDD: SIGKDD Explorations: Newsletter of the Special Interest Group (SIG) on Knowledge Discovery & Data Mining, ACM*, 2, 2000.

[25] R. Cooley. Web Usage Mining: Discovery and Application of Interesting Patterns from Web Data, 2000.

[26] R. Cooley, J. Srivastava, and B. Mobasher. Web Mining: Information and Pattern Discovery on the World Wide Web. In *Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'97)*, November 1997.

[27] Hiroki Arimura, Atsushi Wataki, Ryoichi Fujino, Shinichi Shimozono, and Setsuo Arikawa. An Efficient Tool for Discovering Simple Combinatorial Patterns from Large Text Databases. In *Discovery Science*, pages 393–394, 1998.

[28] Hiroki Arimura, Jun ichiro Abe, Hiroshi Sakamoto, Setsuo Arikawa, Ryoichi Fujino, and Shinichi Shimozono. Text Data Mining: Discovery of Important Keywords in the Cyberspace. In *Kyoto International Conference on Digital Libraries*, pages 121–126, 2000.

[29] H. Ahonen, O. Heinonen, M. Klemettinen, and I. Verkamo. Applying Data Mining Techniques in Text Analysis. Technical Report C-1997-23, 1997.

[30] Carlos Ordonez, Cesar A. Santana, and Levien de Braal. Discovering Interesting Association Rules in Medical Data. In *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 78–85, 2000.

[31] Ahmed Y. Tawfik, , and Krista Strickland. Mining Medical Data for Causal and Temporal Patterns, 2000.

[32] Carlos Ordonez, Edward Omiecinski, Levien de Braal, Cesar A. Santana, Norberto Ezquerra, Jose A. Taboada, David Cooke, Elizabeth Krawczynska, and Ernest V. Garcia. Mining Constrained Association Rules to Predict Heart Disease. In *ICDM*, pages 433–440, 2001.

[33] J. Pfaltz and C. Taylor. Closed Set Mining of Biological Data, 2002.

[34] Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. Knowledge Discovery and Data Mining: Towards a Unifying Framework. In *Knowledge Discovery and Data Mining*, pages 82–88, 1996.

[35] A. Freitas and S. Lavington. Using SQL Primitives and Parallel db Servers to Speed up Knowledge Discovery in Large Relational Databases, 1996.

[36] Marcel Holsheimer, Martin L. Kersten, Heikki Mannila, and Hannu Toivonen. A Perspective on Databases and Data Mining. In *128*, page 10. Centrum voor Wiskunde en Informatica (CWI), ISSN 0169-118X, 30 1995.

[37] K. Morik and P. Brockhausen. A Multistrategy Approach to Relational Knowledge Discovery in Databases. In *Proceedings of the 3nd International Workshop on Multistrategy Learning*, pages 17–28. AAAI Press, 1996.

[38] Proceedings of the First International Workshop on Multi-Relational Data Mining.

[39] S. Muggleton. Inductive Logic Programming. In *The MIT Encyclopedia of the Cognitive Sciences (MITECS)*. MIT Press, 1999.

[40] H. Blockeel and L. De Raedt. Relational Knowledge Discovery in Databases. In S. Muggleton, editor, *Proceedings of the 6th International Workshop on Inductive Logic Programming*, pages 1–13. Stockholm University, Royal Institute of Technology, 1996.

[41] Saso Dzeroski. Multi-relational Data Mining: An Introduction. *SIGKDD Explor. Newsl.*, 5(1):1–16, 2003.

[42] S. Muggleton. Inverse Entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3-4):245–286, 1995.

[43] Stefan Kramer and Gerhard Widmer. Inducing Classification and Regression Trees in First Order Logic. pages 140–159. September 2001.

[44] H. Blockeel and L. De Raedt. Top-down Induction of Logical Decision Trees, 1997.

[45] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, Santa Mateo, 1988.

[46] D. Heckerman. A Tutorial on Learning with Bayesian Networks, 1995.

[47] David Heckerman, Dan Geiger, and David Maxwell Chickering. Learning Bayesian networks: The Combination of Knowledge and Statistical Data. In *KDD Workshop*, pages 85–96, 1994.

[48] W. Buntine. A Guide to the Literature on Learning Probabilistic Networks from Data. *IEEE Trans. On Knowledge And Data Engineering*, 8:195–210, 1996.

[49] Manfred Jaeger. Relational Bayesian Networks. In Morgan Kaufmann, editor, *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence*, pages 266–273, 1997.

[50] Daphne Koller and Avi Pfeffer. Probabilistic Frame-Based Systems. In *AAAI/IAAI*, pages 580–587, 1998.

[51] Lise Getoor. Learning Probabilistic Relational Models. *Lecture Notes in Computer Science*, 1864:322–??, 2000.

[52] Lise Getoor, Nir Friedman, Daphne Koller, and Benjamin Taskar. Learning Probabilistic Models of Relational Structure. In *Proc. 18th International Conf. on Machine Learning*, pages 170–177. Morgan Kaufmann, San Francisco, CA, 2001.

[53] Hendrik Blockeel and Werner Uwents. Using Neural Networks for Relational Learning. In *Proceedings of ICML-2004 workshop on Statistical Relational Learning and its Connections to Other Fields, Banff, Canada*, 2004.

[54] Ming-Syan Chen, Jiawei Han, and Philip S. Yu. Data mining: An Overview from a Database Perspective. *IEEE Trans. On Knowledge And Data Engineering*, 8:866–883, 1996.

[55] Arno J. Knobbe, Arno Siebes, and Daniel van der Wallen. Multi-relational Decision Tree Induction. In *Principles of Data Mining and Knowledge Discovery*, pages 378–383, 1999.

[56] Hctor Ariel Leiva. MRDTL: A Multi-relational Decision Tree Learning Algorithm, 2002.

[57] J. Ross Quinlan and R. Mike Cameron-Jones. FOIL: A Midterm Report. In *Machine Learning: ECML-93, European Conference on Machine Learning, Proceedings*, volume 667, pages 3–20. Springer-Verlag, 1993.

[58] J. Ross Quinlan and R. Mike Cameron-Jones. Induction of Logic Programs: FOIL and Related Systems. *New Generation Computing*, 13(3&4):287–312, 1995.

[59] Nico Brandt, Peter Brockhausen, Marc de Haas, Jrg-Uwe Kietz, and et al. Mining Multi-Relational Data, Deliverable D15, 2001.

[60] Thure Etzold Mahmut Uludağ, Mehmet R. Tolun. A Multi-relational Rule Discovery System. In *ISCIS2003*, pages 252–259, 2003.

[61] Sever H. Uludağ M. Tolun, M.R. and S.M. Abu-Soud. ILA-2: An Inductive Learning Algorithm for Knowledge Discovery. In *Cybernetics and Systems: An International Journal*, volume 30, pages 609–628, 2003.

[62] U. M. Fayyad and K. B. Irani. Multi-interval Discretization of Continuous-valued Attributes for Classification Learning. In *Proc. of the 13th IJCAI*, pages 1022–1027, Chambery, France, 1993.

[63] James Dougherty, Ron Kohavi, and Mehran Sahami. Supervised and Unsupervised Discretization of Continuous Features. In *International Conference on Machine Learning*, pages 194–202, 1995.

[64] Ramakrishnan Srikant and Rakesh Agrawal. Mining Quantitative Association Rules in Large Relational Tables. In H. V. Jagadish and Inderpal Singh Mumick, editors, *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 1–12, Montreal, Quebec, Canada, 4–6 June 1996.

[65] J. R. Quinlan. Unknown Attribute Values in Induction, July 12 1998.

[66] Margaret H. Dunham. *Data Mining: Introductory and Advanced Topics*. Prentice Hall PTR, 2002.

[67] I. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes, and S. Cunningham. Weka: Practical Machine Learning Tools and Techniques with Java Implementations, 1999.

[68] Rakesh Agrawal and Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 12–15 1994.

[69] Leo Breiman. Bagging Predictors. *Machine Learning*, 24(2):123–140, 1996.

[70] Pedro Domingos. Why Does Bagging Work? A Bayesian Account and its Implications. In *Knowledge Discovery and Data Mining*, pages 155–158, 1997.

[71] J. Ross Quinlan. Bagging, Boosting, and C4.5. In *AAAI/IAAI, Vol. 1*, pages 725–730, 1996.

[72] Eric Bauer and Ron Kohavi. An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. *Machine Learning*, 36(1-2):105–139, 1999.

[73] Jie Cheng, Christor Hatzis, Hisashi Hayashi, Mark-A. Krogel, Shinichi Morishita, David Page, and Jun Sese. KDD Cup 2001 Report. *SIGKDD Explorations*, 3(2):47–64, 2002.