Survey of Object Oriented Mining for XML Data

T.Sangeetha
PSGR Krishnammal College for Women
Coimbatore

G.Sophia Reena
HOD – Dept of BCA
PSGR Krishnammal College for Women
Coimbatore

T.Priya
TamilNadu College of Engineering
Coimbatore

ABSTRACT
According to the Petr Kuba, to adapt OR-FP for mining in XML data we preserve basic principles of the algorithm and modify only the input interface. To map XML data to our system we can use the following mapping: XML elements can be processed similarly to the objects in object-oriented data. The name of element corresponds to the class and the attributes of element correspond to the attributes of object. The content of the element (text nodes and elements) can be stored in a special attribute of the object. The type of this attribute should be a set or list – depending on whether we want to deal with an order of nodes. Some specifications (XPointer, XLink) add one more interesting feature to XML data – they allow us to use references to another documents or elements. We can represent this relation as simply as object references. Our proposal is to mining frequent pattern in collection of XML documents.

General Terms
Frequent Pattern,XML,XQuery,XQSharp.

Keywords
Object oriented data mining, OR-FP

1. MOTIVATION
Traditionally data mining methods are employed in various data stored in traditional database management systems and even to relational database management systems. So here comes the comparative study between the former and the latter.

When we compare traditional database management systems with object oriented database management system the following are the important factors that are to be resolved. Object data model solve many limitations of RDBMS. It is better at modeling complex objects and also better in performance on certain data structures. No impedance mismatch between data access language (declarative SQL) and host language (procedural C or Java).

The Object-oriented approach overcomes the no set-valued attributes which is one of the shortcomings. Moreover traditional DBMS is query triggered data exploration whereas the data mining technology is automatic data exploration. Secondly we formulate a hypothesis and test it by sifting through the database for the former part whereas one of the important task of data mining technology is hypothesis testing.

One important factor which motivates to work on this paper is one must know the exact information we are seeking for in the case of traditional DBMS. But vague information is possible to know the correlations or patterns.

Hence OODBMS are capable of storing complex objects i.e., Objects that are composed of other objects and/or multi-valued attributes, object oriented data models are often inspired by OO programming languages C++,C# etc and in worldwide market OODBMS product records about $30 million. Thus the impact of Object-oriented DBMS paves a way to employ the mining methods to mine the frequent patterns in object oriented data.

2. INTRODUCTION

2.1 Conceptual Theory on Discussion

The Theoretical background of the decision tree, clustering and other statistical techniques fall into this category.

Fundamental Dependencies – Large volume of database schema, to predict the occurrence of frequent itemset since it increases the reliability and robustness. It also reduces the complexity of handling the data, since the frequent itemset can predict the customer or client's nature in real-world application.

2.2 Frequent Patterns

The concept of frequent pattern was formulated by Agrawal et al in 1993. Mining frequent pattern is a pattern that covers at least a maximum number of objects. A very good example is Market – Basket problem which helps to find the future customer behavior. The aim of frequent pattern is to find all the frequent itemsets, those patterns that appear in at least a given percentage of all transactions, in other words to find the relation the set of items in a large collection of transactions.

2.2.1 Why frequent pattern growth is fast?
According to the performance study, frequent pattern growth is an order of magnitude faster than Apriori and is also faster than tree-projection.
Reason:
- No candidate generation, no candidate test
- Use compact data structure
- Eliminate repeated database scan
- Basic operation is counting and FP-tree building

2.2.2 Frequent Set
Definition - Let T be the transaction database and σ be the user-specified minimum support. An itemset X A is said to be a frequent itemset in T with respect to σ, if $S(X)_T \geq \sigma$

Example - Let us consider the following set of transactions in a bookshop.

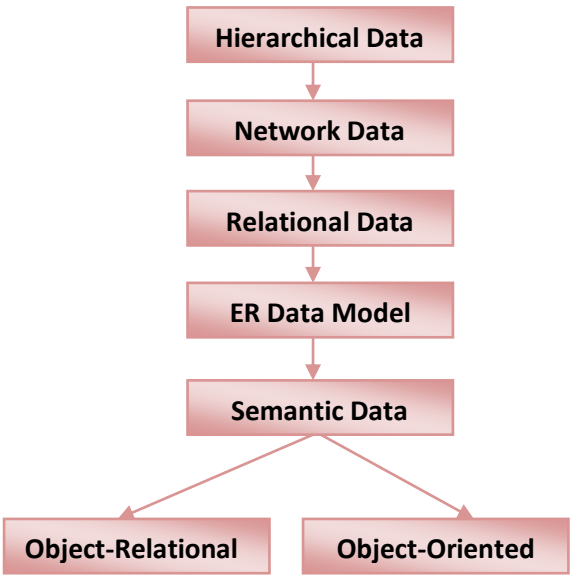
Solution: The purchases (transactions) are made of books on Compiler Construction, Databases, Theory of Computations, Computer Graphics, and Neural Networks. We shall denote these subjects by CC, D, TC, CG and ANN respectively.

t_1:={ANN,CC,TC,CG}
t_2:={CC,D,CG}
t_3:={ANN,CC,TC,CG}
t_4:={ANN,CC,TC,CG}
t_5:={ANN,CC,D,TC,CG}
t_6:={CC,D,TC}

If we assume σ = 50%, then {ANN, CC, TC} is a frequent set as it is supported by at least 3 out of 6 transactions. We can see that any subset of this set is also a frequent set. On the other hand, {ANN, CC, D} is not a frequent set and hence no set which properly contains this set is a frequent set. Discovering all frequent itemsets and their supports is a non-trivial problem if the cardinality of A, the set of items and the database T are large. It should be noted that a large number of itemsets would have minimum support. Even if it is practically feasible, testing support for every possible itemset results in much wasted effort. On the other hand, we have just one counter and make a database pass to count the support for each database. To reduce the combinatorial search space, all algorithms exploit the two properties like downward closure property and upward closure property. Thus, this paves the way for doing research in this domain area.

3 COMPARISONS OF DBMS, RDBMS, ORDBMS, AND OODBMS

Consider the database models – transitions.



3.1 Relational DBMS Limitations

- Semantic overloading.
- Poor representation of 'real world' entities
- Poor support for integrity & business constraints.
- Homogeneous data structure.
- Limited operations.
- Difficulty handling recursive queries.
- Difficulty with 'Long Transactions'
- The many - many relationships is difficult to express.
- The RDBMS has domains, keys, multi-valued and join dependencies.
- Normalisation (Normal forms and FDs) sometimes lead to relations which do not exist or correspond to entities in the real world. This compound on the 'join' feature of query processing.

3.2 Advantages of OODBMS

-Enriched modeling capabilities.
-Extensibility.
-Support for schema evolution.
-Application for advanced database applications.
-Improved performance.

3.3 Disadvantages of OODBMS

-Lack of a universal data model is a big question??
-Ad-hoc querying compromises encapsulation.
-Locking at object-level impacts performance
-Complexity
-Lack of support for views
-Lack of support for security

3.4 What is an ORDBMS?

An Object-Relational database adds features associated with object-oriented systems to a RDBMS (or) Extend the relational data model by including object orientation and constructs to deal with added data types.

3.5 Features of Object – Relational DBMS

OODBS support noted by RDBMS vendors include
-User-extensible type system
-Encapsulation
-Inheritance
-Polymorphism
-Dynamic binding of methods
-Complex objects including first normal form objects
-Object Identity

3.6 Drawbacks of ORDBMS

-Complexity
-Increased costs
-Unclear if the ORDBMS will actually combine relationships and encapsulated objects to correctly and completely mirror the 'real world
-Provision of a language(s)which will front end to SQL and will provide a migration path for existing SQL users

3.7 Comparison of OODBMS / ORDBMS:

OODBMS put more emphasis on the role of the client side. This can improve long, process intensive, transactions.

ORDBMS SQL is still the language for data definition, manipulation and query.

OODBMS have been optimized to directly support object-oriented applications and specific OO languages.

ORDBMS are supported by most of the 'database vendors' in the DBMS market place.

ORDBMS Most third-party database tools are written for the relational model and will therefore be compatible with SQL3

ORDBMS search, access and manipulate complex data types in the database with standard (SQL3 ?), without breaking the rules of the relational data model

OODBMS the ODMG standard group's OQL is now the standard query language amongst OODBMS vendors

3.8 When to use an ODBMS?

In applications that generally retrieve relatively few (generally physically large) highly complex objects and work on them for long periods of time.

3.9 When to use an ORDBMS?

In applications that process a large number of short-lived (generally ad-hoc query) transactions on data items that can be complex in structure.

Concluding Remarks - OODBMS: Abandon SQL (Use an Object Oriented Language instead).ORDBMS: Extend SQL (with Object Oriented feature)

3.10 Object Databases in 2010

The world of data management is changing. The linkage to service platforms, operation within scalable (cloud) platforms, object databases, object-relational bindings, No SQL databases and new approaches to concurrency control are all becoming hot topics both in academia and industry. ODBMS.ORG offers educational resources in all of these new areas.

Object databases (ODBMS) have long been recognized as a solution to one of the biggest dilemmas in modern object-oriented programming (OOP): the object-relational (OR) impedance mismatch. Now that OOP languages like Java and .NET are finally becoming main stream, this problem rests at the heart of information technology.

Thus object databases are increasingly established as a complement to (not a replacement for) relational databases for efficient resolution of the OR mismatch. ODBMSs are flourishing as embeddable persistence solutions in devices, on clients, in packaged software, in real-time control systems, and to power websites.

The open source community has created a new wave of enthusiasm that's now fueling the rapid growth of second-generation, native ODBMSs and demand for appropriate education.

4 STUDY OF ANALYSIS

Consider the classification diagram of data mining algorithms

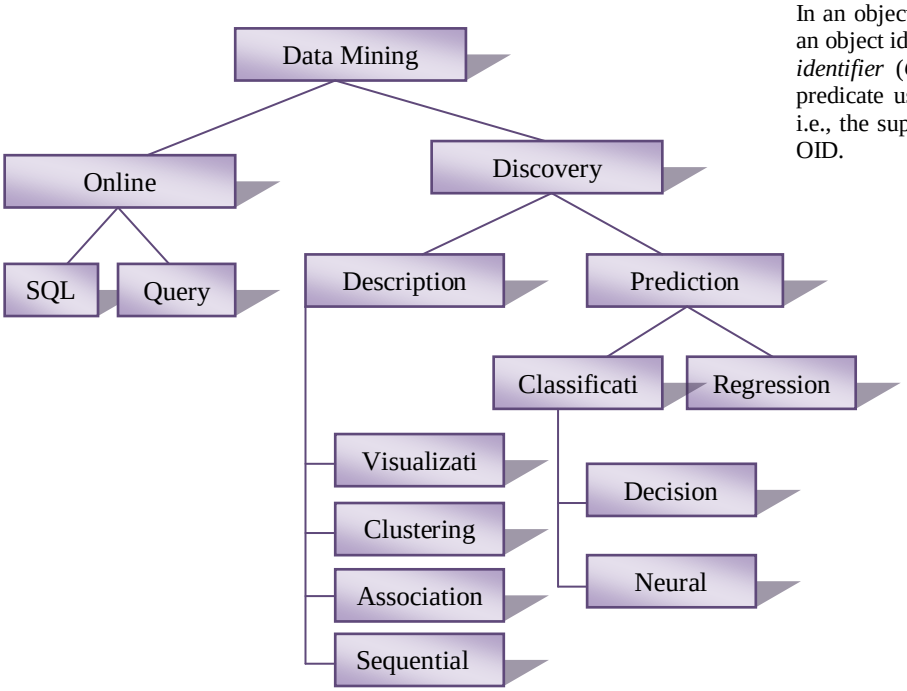


Fig 4.1: Classification of Data Mining Algorithms

According to the data mining algorithm classification, since our discussion is regarding mining the frequent pattern results in association. We consider the discovery driven method under which, description category was chosen because it describes the association which is closely related to our discussion. Now for our comparative study we consider two algorithms like Apriori Algorithm (Base algorithm) and OR-FP Algorithm (Existing algorithm).

4.1 Disadvantages of Apriori Algorithm:

It requires more time.

Large set of candidate key generation.

Requires many database scans.

Large memory requirement.

4.2 Overview of the OR-FP Framework

Farmer and radar showed one of promising directions for further research. By changing data representation or restricting the hypotheses language we can speed up mining process and decrease memory requirements significantly. One could suppose that another possibility is to integrate data mining algorithm with a database systems. The reason is that database systems have long tradition and large effort was given to develop very efficient methods for dealing with large data.

The first system which integrated database technology with an algorithm for mining relational frequent patterns was OR-FP. The OR-FP system extends the Apriori algorithm for mining frequent patterns from a data stored in an *object-oriented* database. Object-oriented databases are powerful enough for representing any data and defining relations among the data and their attributes. It is possible to define a hierarchy of classes, inheritance of objects, or complex data structures as lists and sets.

In an object-oriented database each instance is represented as an object identified by a unique identifier usually called *object identifier* (*OID*). The OID plays the same role as the key predicate used by ILP systems for mining frequent patterns, i.e., the support of a candidate rule is computed according to OID.

OR-FP is the first relational system generating a hypothesis which is not represented in first-order logic. The frequent patterns obtained consist of elements called *fragments*. Fragment is a pair (*X*, *T*) where *X* is a variable representing an object and *T* defines type of that object. The most general pattern, denoted *query class*, is specialized by adding a fragment which introduces another restriction to its subclasses or attributes. OR-FP uses Apriori-like candidate generation and pruning function.

Minsupp

The OR-FP system was implemented in JAVA programming language. It is an application which loads data from Oracle 9i database server2. Because, the database server provides all the meta-data necessary for creating new fragments a user does not have to define any language specification or bias as in the case of ILP systems. This is an indisputable advantage of that system. On the other side, background knowledge cannot be defined. Nonetheless, OR-FP was utilized for solving several benchmark tasks, e.g., propositionalization of Mutagenesis and Carcinogenesis data describing structure of chemical compounds and it beat several well-established systems.

The existing algorithm OR-FP algorithm was introduced as a modification of inductive logic programming (ILP). It is possible to use ILP methods to mine the object oriented or object relational data. However it requires a lot of effort and experience. Thus it paved a way for new algorithm as a modification of ILP or an entirely new algorithm.

This algorithm loads the data from the oracle object relational database system and requires only minimum mandatory settings. OR-FP algorithm and Linear pattern extension algorithm was developed from Apriori algorithm, which is the most popular algorithm to find all the frequent sets. Proportional algorithms are applied to object oriented and thus solving classification and regression tasks in these data. This algorithm was proposed by Peter Kuba and Lubos popelinsky in 2004.

Working Principle

According to this algorithm linear candidate set is generated for each object with minimum support. Counter is incremented for each frequent pattern recognized from the linear candidate set. Support and Minimum Support is measured and compare. If the threshold of support is greater than the threshold of minimum support then the linear frequent pattern and its subsequent candidate set is also generated.

Procedures

Consider an object relational database say OR database, query class C_q, minimum support.

Linear Candidate set LC_1 is generated for each object from the object relational database table.

Counter is incremented for each support.

Comparing the counter values of support and minimum support, linear frequent pattern and non-linear frequent pattern is also generated simultaneously.

Moreover that Linear frequent pattern L_k generated is subjected to the Linear pattern Extension algorithm where the patterns of itemset are classified into three categories like simple, class and attribute type.

Finally if the transaction T_i is a set then the new linear pattern itemset and a new pattern is added to LC_{k+1} set is generated.

The existing algorithm is based on the Apriori algorithm and the same is modified to handle object-oriented data.

Given:

Database : D

Query Class : C_q

: Requested minimum support for the generated frequent patterns. It computes a set of frequent patterns. This is the only input needed.

Factors Involved:

L_i : Linear Frequent Patterns

LC_i : Linear Candidate Patterns

N_i : Non-Linear Frequent Patterns

NC_i : Non-Linear Candidate Patterns

Execution:

At the beginning, exactly one pattern representing the query class is generated in the first step.

Second step, the support corresponds to the number of objects of the query class and its subclasses.

By looping the above process, it is extended to all its attributes, subclasses etc.

Output: Linear patterns are generated.

4.3 Advantages of the existing system:

OR-FP Algorithm – Successfully applied for ill-structured data.

Transformation – We do transform data into single table as in Apriori algorithm. In contrary, we assume that the data match a database schema. It is an advantage when focusing on optimization.

Splitting – The labels of nodes cannot be further split into subparts. On approach is more flexible. We can see labels as sets of subparts (Ex. Words)

4.4 Flaws of OR-FP Algorithm

- Cannot handle **continuous data** and it is necessary to discrete continuous attributes. In implementation, Equal Frequency Intervals discretization method is used.
-Cannot handle the **attributes of type list**. However the algorithm for handling the lists is very similar to the set algorithm and therefore this drawback can be resolved.
-Cannot handle **methods in Object Oriented Data**. We could simply include parameter free functions which could be seen as virtual attributes due to returning due to returning some value for each object.

4.5 Comparison of Existing Algorithm (OR - FP) and Apriori Algorithm

According to this study a new algorithm called OR-FP algorithm was developed as a modification of Inductive Logic Programming (ILP) and also derived the features of Apriori

algorithm which is one of the most popular frequent pattern algorithms. After the considering the issues of mining the object oriented or object relational data requires more effort and experience. They have also developed a linear pattern extension algorithm as an extension of OR-FP algorithm.

According to Apriori algorithm, initially there is a counter for each itemset frequent pattern but there are two counters support and minimum support. The linear frequent pattern itemset is generated if the value of support is greater than minimum support.

According to the former one there is no consideration between linear and non-linear frequent patterns.

Moreover there is no pattern classification in the former algorithm, but the pattern classification is in the latter one. This is the important difference between Apriori algorithm and OR-FP algorithm.

5 RESULTS AND DISCUSSION

5.1 Mining Xml Data

The present dominant data publishing format on the internet is HTML (a markup language that specifies the rendering of the web documents). Recently, XML (eXtensible Markup Language) is gaining popularity as a new standard for data representation and exchange on the internet. The web is rich with information. However, the data contained in the web is not well organized which makes obtaining useful information from the web a difficult task. The successful development of XML as a standard to represent semistructured data makes the data contained in the web more readable and the task of mining useful information from the web becomes feasible.

Although tools for mining information from XML data are still in their infancy, they are starting to emerge. As mentioned in Braga, the fast growing amount of available XML data, raises a pressing need for languages and tools to manage collections of XML documents, as well as to mine interesting information from them. There are developments like Xyleme which is a huge warehouse integrating XML data from the web, and also vendors of data management tools such as Microsoft, Oracle and IBM, who have focused on incorporating XML technologies in their products.

Therefore, it is essential that direct techniques for mining XML data are developed. The query language XQuery was proposed by the W3C and is currently in “last call” status. The purpose of XQuery is to provide a flexible way to extract XML data and provide the necessary interaction between the web world and database world.

XQuery is expected to become the standard query language for extracting XML data from XML documents. Therefore, if we can mine XML data using XQuery, then we can integrate the data mining technique into XML native databases. So, we are interested to know whether XQuery is expressive enough to mine XML data. One data mining technique that has proved popular is association rule mining.

It finds associations between items in a database. In this study we show that XML data can be mined using XQSharp (similar to XQuery, which is proposed by Microsoft) and discuss the XQSharp implementation of the OR-FP algorithm. Moreover, we discuss other useful capabilities that need to be added into XQSharp to make association rule mining efficient.

6. IMPLEMENTATION

The following are the implementation screens for our object oriented xml mining.

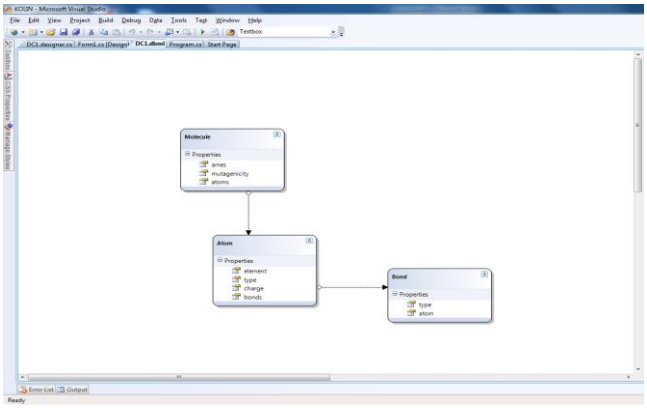


Fig 6.1: Object Relational Mapping

The figure 6.1 shows the object relational mapping for the mutagenesis dataset. In our current implementation will support only the data which have been configured in application.

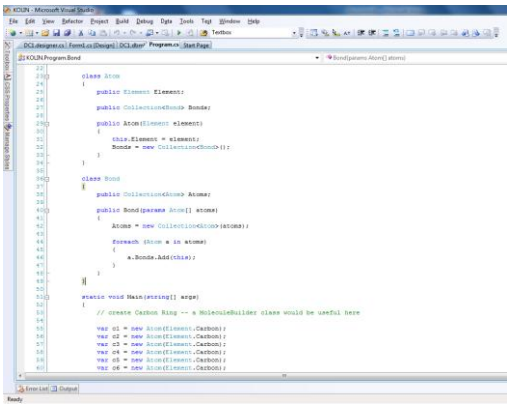


Fig 6.2: Internal Object Oriented Data Structure

The above figure 6.2 shows the design pattern of our implementation part. It shows the object oriented design for our mutagenesis dataset. This implementation is considered as proof of concept. But still needs to be enhanced to handle the ad hoc document. The xml structure is opted to specify the data structure of the dataset.

In our implementation, we have given two options. The first option is synthetic data and real data. For a trial basis simple random value is generated using the synthetic data. The real data is an option to handle real-time dataset, which should set before as xml pattern.

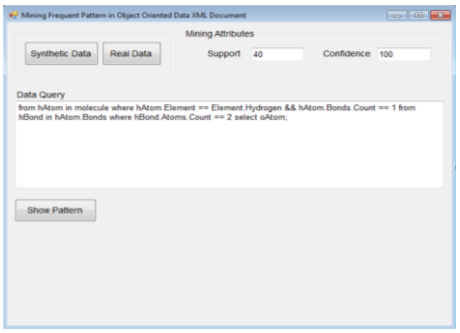


Fig 6.3 User Interface for Object Oriented Mining XML Data

In the user interface user can specify the support and confidence percentage. Further they need to give the data query as given in the figure 6.3. The show pattern button executes the process and display frequency pattern.

Time measures have been taken to for the synthetic data towards minimum support. Our experiment is executed with windows 7 operating system, Intel Core 2 Duo Processor, 4 GB RAM, 500 GB HDD. It is performed on XQSharp engine inside the native XML database support.

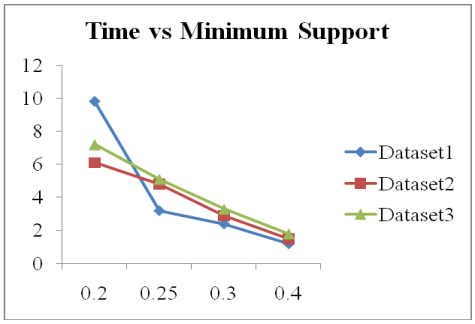


Fig 6.4 Chart between Time and Minimum Support for different dataset

We see that the performance of the XQSharp implementation is dependent on the number of large itemsets found and the size of the dataset. For example, the number of large itemsets found from dataset-1 is much more than dataset-2 and dataset-3 with minimum support 20%. The running time for dataset-1 with minimum support 20% is much higher than the running time of dataset-2 and dataset-3, since the number of large itemsets found for dataset-1 is about 2.4 times more than the other datasets.

We also notice that the majority of the time for the XQSharp implementation is spent in counting support to find large itemsets. The XQuery implementation requires that for each itemset in the Candidate set, it will read the database once to obtain the support value. Therefore, the number of times needed to scan the database to obtain the support count for finding large itemsets is $O(2^l)$, where l is the length of the maximal large itemset.

7. CONCLUSION AND SCOPE FOR FUTURE WORK

This study reveals that an implementation of the object oriented mining for xml data. According to the Petr Kuba, to adapt OR-FP for mining in XML data we preserve basic principles of the algorithm and modify only the input interface. To map XML data to our system we can use the following mapping: XML elements can be processed similarly to the objects in object-oriented data. The name of element corresponds to the class and the attributes of element correspond to the attributes of object.

The implementation is developed using Visual Studio 2008 in C# with XQSharp and LINQ. The present implementation is executed as proof of concept basis. But still it can be enhance the same idea to apply for any type of dataset. The query part in this study implementation is more like programmers query, it is also be further optimized into user preferable pattern.

We ask that authors follow some simple guidelines. In essence, we ask you to make your paper look exactly like this document. The easiest way to do this is simply to download the template, and replace the content with your own material.

8. ACKNOWLEDGMENTS

Our sincere thanks to Dr. M. S. Vijaya M.Sc., M.Phil., Ph.D, Head of the Department, Department of Computer Science (PG), P.S.G.R.Krishnammal College for Women, Coimbatore for her valuable guidance and suggestions during the course of study.

9. REFERENCES

[1] Fayyad, U. M., Piatetsky-Shapiro, G., and Smyth, P. From Data Mining to Knowledge Discovery in Databases. *AI Magazine 17*, 3 (1996), 37–54.

[2] Famili, A., Shen, W.-M., Weber, R., and Simoudis, E. Data preprocessing and intelligent data analysis. *Intelligent Data Analysis Journal 1*, 1 (1997).

[3] Kramer, S., Lavraˇc, N., and Flach, P. Propositionalization approaches to relational data mining. In *Relational Data Mining*, S. Dˇzeroski and N. Lavraˇc, Eds. Springer-Verlag, September 2001, pp. 262–291.

[4] Domingos, P. Unifying instance-based and rule-based induction. *Machine Learning 24*, 2 (1996), 141–168.

[5] Agrawal, R., Imielinski, T., and Swami, A. N. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26–28, 1993* (1993), P. Buneman and S. Jajodia, Eds., ACM Press, pp. 207–216.

[6] Agrawal, R., and Srikant, R. Fast algorithms for mining association rules in large databases. In *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12–15, 1994, Santiago de Chile, Chile* (1994), J. B. Bocca, M. Jarke, and C. Zaniolo, Eds., Morgan Kaufmann, pp. 487–499.

[7] Muggleton, S. Inductive Logic Programming. *New Generation Computing 8*, 4 (1991), 295–318.

[8] Muggleton, S., and Raedt, L. D. Inductive logic programming: Theory and methods. *Journal of Logic Programming 19/20* (1994), 629–679.

[9] Dehaspe, L., and Toivonen, H. Frequent query discovery: a unifying ILP approach to association

rule mining. Tech. Rep. CW 258, Katholieke Univesiteit Leuven, Departmen of Computer Science, Celestijnenlaan 200A – B-3001 Heverlee (Belgium), March 1998.

[10] Fayyad, U. M., Piatetsky-Shapiro, G., and Smyth, P. Knowledge Discovery and Data Mining: Towards a Unifying Framework. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)* (1996), p. 82.

[11] Mannila, H., and Toivonen, H. An algorithm for finding all interesting sentences. In *Proceedings of the 6th Internation Conference on Database Theory* (1996), pp. 215–229.

[12] Dehaspe, L., and Raedt, L. D. Mining association rules in multiple relations. In *ILP* (1997), N. Lavrac and S. Dzeroski, Eds., vol. 1297 of *Lecture Notes in Computer Science*, Springer, pp. 125–132.