POSITIVE AND NEGATIVE ASSOCIATION RULE MINING ON XML DATA STREAM IN DATABASE AS A SERVICE CONCEPT

by

Samet Çokpınar

B.Sc., in Computer Engineering, Marmara University, 2006

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering
Boğaziçi University
2011

ACKNOWLEDGEMENTS

First, I would like to thank my advisor, Professor Taflan İmre Gündem for his guidance and support during the development of this research. His patience and understanding helped me solve the problems and continue with the work harder.

I would also like to thank Associate Professor Can Özturan and and Assistant Professor Mustafa Ağaoğlu for kindly accepting to be in my thesis jury.

I give my best regards and special thanks to my wife Nevin Çokpınar. She was always my key supporter throughout my graduate studies and particularly this research. Without her endless support and encouragement, it would have been so hard to pass through all the hard times.

I also would like to thank to my family for their endless support in my whole education life.

Lastly, I would like to thank to my dear friend Melih Çelik and my manager İsa Ergül for their support to finish this thesis.

ABSTRACT

POSITIVE AND NEGATIVE ASSOCIATION RULE MINING ON XML DATA STREAM IN DATABASE AS A SERVICE CONCEPT

Due to the development of database technology and systems in recent years, there is an enormous increase in data size. This increase makes the data mining one of the hot topic for organizations to determine their strategies. Association rule mining is a data mining approach that discovers the useful, but hidden patterns in the data set. This method uses widely in traditional databases and usually to find the positive association rules. However, there are some other challenging rule mining topics like data stream mining and negative association rule mining. Nowadays, organizations want to concentrate on their own business and outsource the rest of their work. This approach reveals the "Database as a service" concept. This concept provides lots of benefits to data owner, but, at the same time brings out some security problems.

In this research, we have proposed a mining model that combines the mentioned challenging areas. To the best of our knowledge, our approach is unique in the literature. Our model provides efficient solution to find positive and negative association rules on XML data stream in database as a service concept. We have adapted some pruning strategies for efficient negative rule mining. Also, we have applied some security techniques to provide efficient and sufficient data protection.

We have run many experiments with some different synthetic data sets and with one real world data set to show the efficiency of our proposed model. The results have shown that proposed system makes the association rules mining operation efficiently.

ÖZET

XML VERİ KATARLARINDA POZİTİF VE NEGATİF BİRLİKTELİK KURALLARININ ÇIKARIMI

Son yıllarda veritabanı teknolojisindeki gelişmeler, saklanan veri miktarlarında önemli bir artışa yol açmıştır. Bu artış veri madenciliğini şirketlerin stratejilerini belirlemeleri açısından önemli bir konu haline getirmiştir. Birliktelik kuralları çıkarımı, veri kümeleri içerisindeki yararlı ama gizli olan örüntüleri ortaya çıkaran bir veri madenciliği yaklaşımıdır. Bu metot geleneksel veritabanları üzerinde pozitif birliktelik kurallarının çıkarımında yaygın olarak kullanılmaktadır. Fakat, bu konuda veri katarı madenciliği ve negatif birliktelik kuralları çıkarımı gibi daha zor problemler de yer almaktadır. Günümüzde şirketlerin büyük kısmı kendi uzmanlık alanlarına odaklanmak ve diğer işlerinin servis sağlayıcılar tarafından yapılmasını istemektedirler. Bu yaklaşım ”dış kaynaklı veritabanı” konseptini ortaya çıkarmıştır. Bu konsept veri sahiplerine birçok fayda sağlarken aynı zamanda bazı güvenlik problemlerini de ortaya çıkarmaktadır.

Biz bu çalışmada yukarıda bahsetmiş olduğumuz bu zor problemlerin çözümlerini bir araya getiren bir madencilik modeli önerdik. Literatürde bizim önerdiğimiz şekilde bir yaklaşıma rastlanmamıştır. Bizim modelimiz XML veri katarlarında pozitif ve negatif birliktelik kuralları çıkarımı işlemini dış kaynaklı veritabanları konsepti ile etkin bir şekilde gerçekleştirmektedir. Bu çalışmada verimli bir negatif birliktelik kuralı çıkarımı için bazı eleme teknikleri kullanılmıştır. Ayrıca etkin ve yeterli bir veri koruması için bazı güvenlik teknikleri kullanılmıştır.

Bu çalışmada önerilen modelin verimliliğini göstermek amacıyla farklı veri kümeleriyle birçok test yapılmıştır. Test sonuçları önerilen modelin veri kuralları çıkarımı işlemini etkin bir şekilde yaptığını göstermiştir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . iii
ABSTRACT . . . iv
ÖZET . . . v
LIST OF FIGURES . . . viii
LIST OF TABLES . . . ix
LIST OF SYMBOLS/ABBREVIATIONS . . . x
1. INTRODUCTION . . . 1
 1.1. Introduction . . . 1
 1.2. Outline . . . 3
2. PRELIMINARIES . . . 5
 2.1. Association Rule Mining Definitions & Concepts . . . 5
  2.1.1. Apriori Algorithm . . . 9
  2.1.2. FP-Growth Algorithm . . . 11
 2.2. Association Rule Mining on Data Streams . . . 14
 2.3. Database as a Service Concept . . . 16
 2.4. Encryption & Security Issues . . . 16
3. ARCHITECTURE OF THE PROPOSED SYSTEM . . . 19
 3.1. Problem Statement . . . 19
 3.2. Proposed Model . . . 20
  3.2.1. Model Assumptions . . . 20
  3.2.2. PNRMXS Algorithm . . . 20
4. EXPERIMENTAL RESULTS . . . 34
 4.1. Experimental Data Sets . . . 34
  4.1.1. IBM Synthetic Data Set Generator . . . 34
 4.2. Experimental Environment . . . 36
 4.3. Experimental Results . . . 36
  4.3.1. FP-Tree Node Counts . . . 36
  4.3.2. FP-Tree Generation Time Comparison . . . 37
  4.3.3. Total Execution Time Comparison . . . 39

4.3.4. Execution Time Comparison of PNRMXS & Moment Algorithm 41
4.3.5. Memory Consumption Comparison 44
4.3.6. Effect of Block Size on Execution Time 45
4.3.7. Negative Rule Mining Effect 46
4.3.8. Example Rule Set 47
5. DISCUSSIONS & CONCLUSION 49
6. FUTURE WORKS 51
REFERENCES 53

LIST OF FIGURES

Figure 2.1. Apriori Algorithm Pseudo-Code by R. Agrawal [2] 10

Figure 2.2. Apriori-Gen Procedure Pseudo-Code by R. Agrawal [2] 10

Figure 2.3. FP-Tree Construction by P. N. Tan [5] 11

Figure 2.4. FP-Growth steps to find frequent sets ending with $\{e\}$ by P. N. Tan [5] 13

Figure 3.1. Flowchart of the PNRMXS Algorithm 22

Figure 3.2. Example XML Data 23

Figure 3.3. Flowchart of Stream Encryption & Decryption Processes 25

Figure 3.4. PNRMXS Algorithm Pseudo-Code 30

Figure 4.1. FP-Tree Node Counts 37

Figure 4.2. FP-Tree Generation Time Comparison 38

Figure 4.3. Moment & PNRMXS Comparison 43

Figure 4.4. Execution Time with Different Block Size 45

Figure 4.5. Negative Rule Mining Effect on Execution Time 46

Figure 4.6. Example Rule Set 48

LIST OF TABLES

Table 2.1. Example Market Basket Data Set by P. N. Tan [5] 7

Table 3.1. 2x2 Contingency Table for Binary Variables by M. L. Antonie [13] 28

Table 4.1. Descriptions of Parameters 35

Table 4.2. Generated Test Data Sets 35

Table 4.3. FP-Tree Node Counts in Detail 38

Table 4.4. FP-Tree Generation Times 39

Table 4.5. Execution Times on T5N1000I4D100K data set 39

Table 4.6. Execution Times on T10N1000I4D100K data set 40

Table 4.7. Execution Times on T10N1000I4D100K data set 40

Table 4.8. Execution Time Details 42

Table 4.9. Rule Generation Times on T10N1000I4D100K - T10N2000I4D100K 42

Table 4.10. Memory Consumption Comparison 44

Table 4.11. Memory Consumption for T10N4000I4D100K data set 45

LIST OF SYMBOLS/ABBREVIATIONS

ρ	Correlation Coefficient
σ	Standard Deviation
ϕ	Pearson Correlation Coefficient
AES	Advanced Encryption Standard
$conf\ (X \Rightarrow Y)$	Confidence Value of Item Set
$Cov\ (X,Y)$	Covariance Value
CPU	Central Processing Unit
FP-Growth	Frequent Pattern Tree
FP-Tree	Frequent Pattern Tree
IBM	International Business Machines
I/O	Input and Output
MCN	Minimum Confidence Threshold for Negative Rules
MCP	Minimum Confidence Threshold for Positive Rules
MS	Minimum Support Threshold
MSN	Minimum Support Threshold for Negative Rules
MSP	Minimum Support Threshold for Positive Rules
PNRMXS	Positive and Negative Rule Mining on XML Data Stream
RSA	Rivest, Shamir and Adleman
$supp\ (X \Rightarrow Y)$	Support Value of Item Set
W3C	The World Wide Web Consortium
XML	eXtensible Markup Language

1. INTRODUCTION

1.1. Introduction

Due to the development of database technology and systems in recent years, there is an enormous increase in data size. This increase makes the data mining, also called knowledge discovery in databases, technology one of the hot topic in some business domains like marketing, WEB technologies, financing and telecommunication. Data mining is an approach that discovers the useful, but hidden patterns within the huge data. These hidden patterns can provide to us some interesting and valuable relationships or associations. The process of finding these relationships is named as Association Rule Mining.

Association rule mining was first introduced by Agrawal et al. [1] in order to mine association rules on large transactional databases. After this concept, Agrawal and Srikant [2] have developed the most popular association rule mining algorithm Apriori that is based on minimum support and confidence constraints. This algorithm is easy to implement and can be used on databases that do not change often. However, the main drawback of this algorithm is slowness that is due to the lots of passes over the data set. Therefore, another and one of the fastest rule mining algorithm, FP-Growth (Frequent Pattern Growth), is proposed by Han et al. [3]. There are two main improvements in this algorithm. First, FP-Tree is the compressed form of the database, so the data size of the tree is usually smaller than the original data size that provides efficient memory usage. The other improvement is that there is no candidate set generation in FP-Growth algorithm that is the main reason of fastness of it.

Association rule mining, generally, is understood as positive association rule mining. Positive association rule is stated as "if A occurs in a transaction, then B will likely also occurs in the same transaction" in [4]. However, with the increasing usage of data mining technology, researches has recently focused on finding alternative patterns like negative correlated patterns [4, 11, 13, 14]. In a negative correlated pattern, we expect that there should be a negative relation between the item sets. The example given in [4] summarizes a negative rule very well; "birds can fly is a well-known fact, but penguins can not fly although they are birds." There is a few algorithms are proposed in the literature about the negative rule mining because of it's novelty and difficulty.

Nowadays, in lots of applications like network measurements, telecommunications data transmission is made with data streams. This wide usage increases the importance of data mining topic on data streams. Data streams arrive continuously with high speed and contains huge amount of data, so fast analyzing and processing of the data is a crucial point. Because of these features of data streams, mining process on data streams is more difficult than mining static data.

In the literature, there are many proposed rule mining methodologies that are based on Apriori algorithm [12, 17, 18]. However they are not applicable on data streams, because these methods need iterative scans over the data set and generate large number of candidate frequent item sets. Also, original FP-Growth algorithm is not suitable for data stream mining, because it makes two scans over data, which is not possible in a streaming environment. Therefore, researchers focus to develop single-pass mining algorithms [16, 19, 27].

Data mining on XML structured data is another challenging research area. XML (eXtensible Markup Language) is a W3C recommended markup language to transmit and store the data in a structure way. Increasing in usage of internet makes the XML one of the most used data interchanging format. Therefore, data mining on XML data becomes an important topic. Mining process on XML data can be in different ways. First, the XML data can be converted to another format like flat file or other relation structure and after that mining algorithms can be run over the converted format. However, the processing techniques in this method should be chosen carefully without affecting the mining task negatively. Another rule mining technique on XML data is working on native XML without making any conversion. For this aim, a lot of researchers use XML query language XQuery [42]. The main aim of XQuery is to query XML data, so implementation of complex algorithm is a difficult task. The Apriori algorithms is implemented by XQuery [32]. In [34], authors implement the FP-Growth algorithm with XQuery language. However, in both implementations, authors did not give any experimental result about the performance of their implementations. Another study about XML data stream mining with XQuery is made by Jacky et. al in [46]. In this research, some performance tests are made on Apriori implementation with XQuery. According to their results, XQuery is more suitable for mining data from small data sets.

Today, organizations want to concentrate on their own professions and they prefer to outsource the rest of their work. For instance, efficient data processing is a fundamental and vital issue for almost every organization. However, managing the large database systems is an expensive and complicated task. Therefore, organization want to delegate database processes like administration, backup, migration and optimization to a service provider and that concept is called as "database as a service". On the other hand this concept arises some problems about the security of the data, because data owners share their private information with the third parties.

The aim of this research is to propose a model that can solve the problem of combination of the mentioned research areas in the previous paragraphs in one model and we call it PNRMXS (Positive and Negative Association Rule Mining on XML Data Stream). In order to implement our proposed model, we have made some adaptations on the algorithms that we use them as a basis model.

1.2. Outline

The rest of this document is organized as follows:

Chapter 2 will give the background information about the association rule mining techniques and models that will be used in this research.

In Chapter 3, first our problem statement will be described. Also, we will give all details of our proposed technique.

Chapter 4 will explain the experimental setups that are used to validate our proposed model and also will give the details of the experiments that are performed.

Chapter 5 will summarize our research and present the conclusions driven from the results of this study.

In the last chapter, we will address possible improvements about our proposed model and future works.

2. PRELIMINARIES

In this chapter, we will introduce the background knowledge and related work about the association rules mining topic. Next section will give information about the basic association rule mining concepts and definitions. Also, we will give the details of the two key algorithms in association rule mining topic and also some related works about positive and negative association rule mining. In Section 2.2, we will give the details of the data stream association rule mining concept. Section 2.3 will provide information about the "Database as a Service Concept". In the final section we will give necessary fundamental information about the encryption techniques and security issues.

2.1. Association Rule Mining Definitions & Concepts

Association rule mining is one of the main techniques of data mining and knowledge discovery. The purpose of association rules is to find frequent patterns, associations and correlation relationships among large set of data items. The most-known, typical and widely-used example of association rule mining is Market Basket Analysis. Market basket analysis has emerged as the next step in the evolution of retail merchandising and promotion. It allows leading retailers to quickly and easily look at the size, contents and value of their customers' market basket to understand the patterns in how products are purchased together or basic product affinities [31]. Besides market basket analysis, association rule mining is also applicable to other domains like medical systems [24, 25, 39], telecommunications [35, 40] and financing [45, 47].

In literature, there are mainly three types of association rules that are binary, quantitative and fuzzy association rules. A binary association rule is a rule that concerns associations between the absence and the presence of items (0 refer for absence and 1 refer for presence of the item). If a rule describes association between quantitative items or attributes, it is named as quantitative association rule. In such rules, if the domain of values for a quantitative approach is large, an obvious approach will be to partition the values into intervals and then map each (attribute, interval) pair to a boolean attribute [22]. After that, any algorithm for finding boolean association rules can be used. In quantitative association rules, if value sets of items in the rule is replaced by fuzzy sets [23], fuzzy association rules is obtained. In this research, we will use the boolean association rules.

The formal definition of positive association rules can be summarized as follows: Let $I = \{i_1, i_2, i_3, ..., i_n\}$ be a set of n distinct literals called items. Let *DB* be a set of transactions, $T = \{t_1, t_2, t_3, ..., t_n\}$,where each transaction is a set of items and is associated with a unique identifier *TID*. Let *S*, called an item set, be a set of items in *I*. The number of items in an item set (S) is the size of an item set (N). A transaction *T* is said to contain *S* if $S \subseteq T$.

There are two sub types of frequent item sets which are maximal frequent item set and closed frequent item set. A maximal frequent item set is an item set which none of its immediate supersets are frequent. An item set X is closed if none of its immediate supersets has the same support count as X and it is frequent if its support value is greater than or equal to user-defined minimum support threshold.

A positive association rule is an implication of the form $X \Rightarrow Y$, where $X \subseteq I$, $Y \subseteq I$ and $X \cap Y = \emptyset$. We call X as the antecedent of the rule, and Y as the consequent of the rule. The strength of an association rule can be measured with it's support and confidence value. The support value of an item set can be defined as the proportion of transactions in the data set which contain the item set. The confidence value of a rule indicates its reliability. The support value is used for frequent item set elimination, while confidence value is used for association rule generation.

$$supp(X \Rightarrow Y) = \frac{(X \cup Y)}{N} \tag{2.1}$$

$$conf(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)} \tag{2.2}$$

In Equation 2.1, $(X \cup Y)$ indicates the total number of transactions that containing the both item X and Y. We use the example that is stated in [5] to understand how the support and confidence framework works.

Table 2.1: Example Market Basket Data Set by P. N. Tan [5]

TID	Items
1	{Bread, Milk}
2	{Bread, Diapers, Beer, Eggs}
3	{Milk, Diapers, Beer, Cola}
4	{Bread, Milk, Diapers, Beer}
5	{Bread, Milk, Diapers, Cola}

In Table 2.1, the binary representation of the market basket data can be seen. This table can be read as follows: *"TID"* column represents the id of current transaction, *"Items"* column represents the list of items is contained in the transaction. For instance; there are 5 transactions in data set and first transaction shows that customers bought *Bread* and *Milk*. For example; consider support and confidence values as 0.3 for data set in Table 2.1. If we mine the data set, we can find {*Milk, Diapers, Beer*} as a frequent set because its support value is 0.4. After mining step, we can generate possible association rules. $\{Milk \Rightarrow Diapers, Beer\}$ is one rule candidate and if we check its confidence that is $2/4 = 0.5$, so it can be stated as a valid positive association rule. The last point we would like to state here is, how the support and confidence threshold values are determined. The values of these constraints depend on the data owners needs and the business domain.

The other type of association rules is the negative association rule. A negative association rule can be defined as "customers that buy product X, but not product Y". Negative associations can provide us valuable information, for example, in marketing strategies. In literature a few researchers tend to solve mining negative association rules problem due to the its difficulty. However, why is the negative rule mining is difficult task? As stated in [11], finding negative associations rule is not easy as positive mining, because search space in negative rule mining is much bigger than positive rule mining.

For example, in a large retail store, there can be tens of thousands of items and also may be billions of customer transactions. For example, if there are 50,000 items, the possible combinations of items is $2^{50,000}$ and even though majority of them will not appear even once in the entire database. As can be seen in this example, there can be millions of negative association rules and unfortunately most of these rules are likely to be meaningless.

Because of the reasons mentioned above, some researchers try to simplify negative association mining problem. For instance, in [14], authors first find a set of positive rules and after that generate negative rules based on existing positive rules and domain knowledge. However, in this strategy, because of using existing positive rules to generate negative rules, you may loose the valuable negative associations. There is another model that is proposed in [15] is based on appending virtual negative items to the transactions. The main drawback of this approach is that there are more processes on the data sets.

The formal definition of negative association rules is similar to positive association rules. The only difference is that in negative association rules,the antecedent or the consequent part of the rule is negated. In negative rule mining, support-confidence framework can also be used and the support and confidence values of the rules are calculated with equations 2.3 and 2.4.

$$supp(X \Rightarrow \neg Y) = supp(X) - supp(X \cup Y) \tag{2.3}$$

$$conf(X \Rightarrow \neg Y) = \frac{supp(X) - supp(X \cup Y)}{supp(X)} \tag{2.4}$$

2.1.1. Apriori Algorithm

The most-known association rule mining algorithm, Apriori, is proposed by Agrawal and Srikant [2]. This model, basically, divides the rule mining process into two sub-problems. In the first step, the algorithm generates the 1 to k large item set where k is the count of separate items in the transactions. After candidate item set generate, algorithm pass over the database to find the frequent large item set that have support value is more than predefined minimum support value. The other step of the algorithm generates association rules from frequent large item sets with minimum confidence constraints. The main advantage of the Apriori algorithm is easy implementation. Also, it can be easily optimized, for instance, it can be parallelized or some parameters other than support and confidence can be added for pruning process. However, there are two bottlenecks of the Apriori algorithm. First one is, it requires too many scans over the database and it leads to high CPU usage and I/O cost. Also, it makes candidate set generation that takes the most of the execution time to find association rules. Because of these bottlenecks, this algorithm is not suitable for data streams, in which data should be scanned only once. The pseudo-code of the Apriori algorithm is provided in [2] and can be seen in Figure 2.1 and Figure 2.2.

```
Algorithm Apriori
1: $L_1 = \{Large\ 1\text{-}Itemset\}$;
2: for $(k = 2;\ L_{k-1} \neq \emptyset;\ k++)$ do
3:    $C_k = apriori\text{-}gen\ (L_{k-1})$; //New Candidates
4:    for all transactions $t \in D$ do
5:       $C_t = subset\ (C_k, t)$ //Candidates contained in t
6:       for all candidates $c \in C_t$ do
7:          $c$.count ++;
8:       end for
9:    end for
10:   $L_k = \{c \in C_k \mid c.\text{count} \geq minsup\}$
11: end for
12: Answer = $\text{U}_k L_k$;
```

Figure 2.1: Apriori Algorithm Pseudo-Code by R. Agrawal [2]

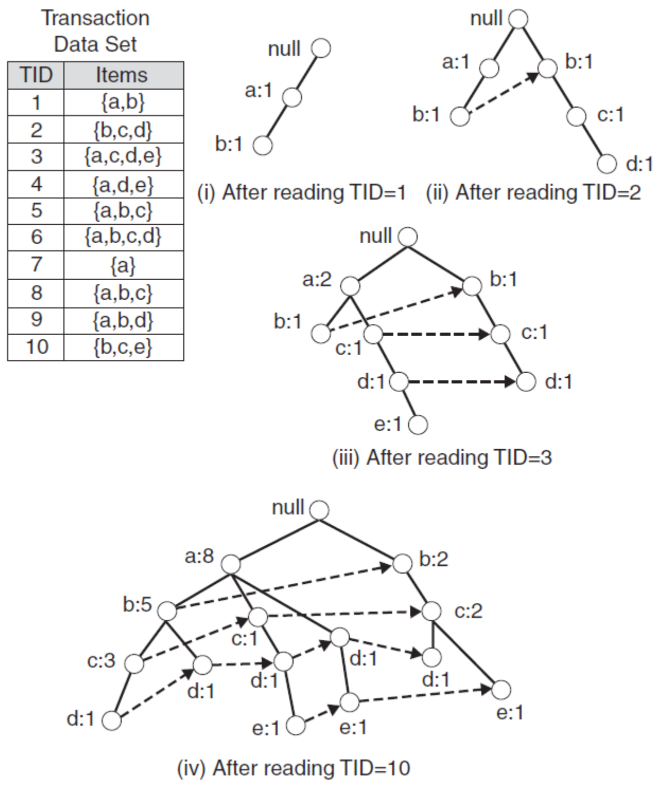
```
Procedure Apriori-Gen
1: insert into $C_k$
2: select $p$.item$_1$, $p$.item$_2$,....,$p$.item$_{k-1}$, $q$.item$_{k-1}$
3: from $L_{k-1}$ $p$, $L_{k-1}$ $q$
4: where $p$.item$_1 = q$.item$_1$,..., $p$.item$_{k-2} = q$.item$_{k-2}$, $p$.item$_{k-1} < q$.item$_{k-1}$;
5: for all itemsets $c \in C_k$ do
6:    for all ($k$-1)-subsets $s$ of $c$ do
7:       if $(s \notin L_{k-1})$ then
8:          delete $c$ from $C_k$;
9:       end if
10:   end for
11: end for
```

Figure 2.2: Apriori-Gen Procedure Pseudo-Code by R. Agrawal [2]

2.1.2. FP-Growth Algorithm

Another popular rule mining algorithm, FP-Growth, is developed by Han et al. [3]. The mining process of the FP-Growth algorithm can be summarized as follows: Initially, the algorithm scans the data set to find the item support value. After that, items that have lower support value than user-specified minimum support value are eliminated. Now, there are transactions which contain only frequent items. In second step, frequent items are put in the FP-Tree. The FP-Tree is a prefix tree that contains frequent items. In the last process, association rules which have higher confidence value that user-specified confidence are mined without candidate set generation that is the main contribution of the FP-Growth algorithm. However, this algorithm makes two scans over data, so the original form of it is not suitable for data stream mining.

How the FP-Growth algorithm runs can be explained on an example that is provided in [5] as follows:



TID	Items
1	{a,b}
2	{b,c,d}
3	{a,c,d,e}
4	{a,d,e}
5	{a,b,c}
6	{a,b,c,d}
7	{a}
8	{a,b,c}
9	{a,b,d}
10	{b,c,e}

Figure 2.3: FP-Tree Construction by P. N. Tan [5]

In Figure 2.3, there is a data set with ten transactions and five items. Each node in the tree contains the label of an item and existence count of item. The root node of the tree depicts with *null* symbol and tree is constructed with following steps:

(i) In order to find the support value of each items, the data set should be scanned firstly. The infrequent items are eliminated, because they do not have importance in mining process. The frequent items are sorted in decreasing support value. In this example data sets, frequent items are $a,\ b,\ c,\ d$ and e .

(ii) In second step, the data set is scanned once more to construct FP-Tree. The first transaction, $\{a,\ b\}$ is read and the nodes with label a and b are created. Also, the support value of each item is set to 1.

(iii) If the transactions do not contain common prefixes, the process is continued as explained in the second step. Otherwise, if there are transactions that contain common items, their paths are overlapped partially or fully in the tree. Because of overlapping paths, support values of common items is increased by 1 and support values of others are set to 1.

(iv) The process is continued until all transactions have been mapped onto the tree. The construction steps of the FP-Tree can be seen in Figure 2.3.

In FP-Growth algorithm, frequent item sets are mined from the FP-Tree by traversing the tree in a bottom-up fashion. In order to understand in detail, how the algorithm works is explained in [5] with the following example. Suppose, we try to find all frequent item sets ending with item $\{e\}$. First, we must compare the support value of item set $\{e\}$ with user-defined support value. If it is frequent, we continue the finding frequent item sets ending with $\{d,\ e\}$, $\{c,\ e\}$ and so on. In each further step, we add new item to the set and check whether new set is frequent or not. Finally, the merging of all result gives us all frequent item sets ending with item $\{e\}$. The details of these processes can be seen in Figure 2.4 and summarized in [5] with following steps:

(i) All the paths containing node $\{e\}$ are found firstly. These paths are named as prefix paths and can be seen in Figure 2.4(a).

(ii) The support value of item $\{e\}$ can be found by following the links between nodes

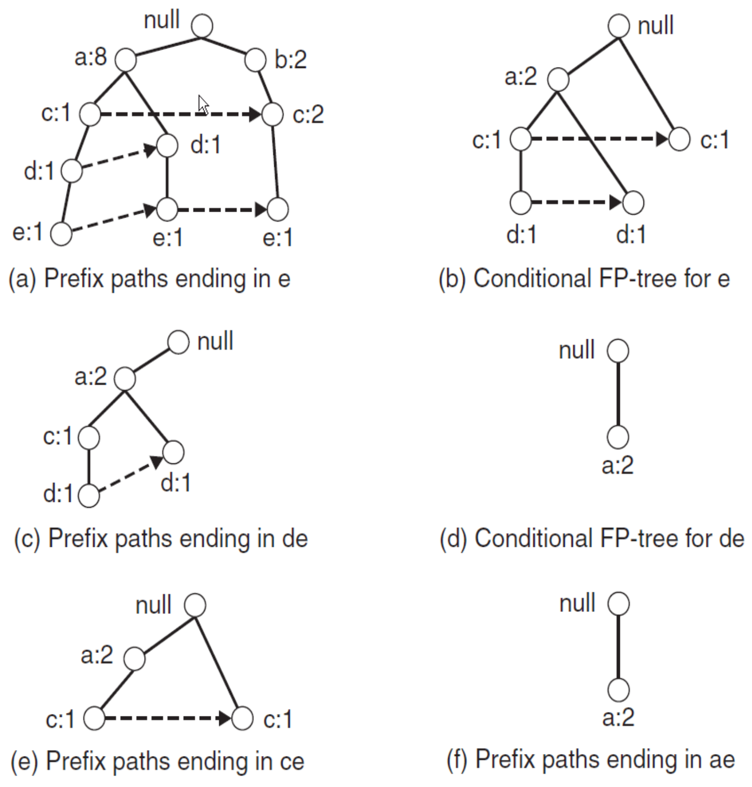

null
a:8
b:2
c:1
c:2
d:1
d:1
e:1
e:1
e:1
(a) Prefix paths ending in e
null
a:2
c:1
c:1
d:1
d:1
(b) Conditional FP-tree for e
null
a:2
c:1
d:1
d:1
(c) Prefix paths ending in de
null
a:2
(d) Conditional FP-tree for de
null
a:2
c:1
c:1
(e) Prefix paths ending in ce
null
a:2
(f) Prefix paths ending in ae

Figure 2.4: FP-Growth steps to find frequent sets ending with $\{e\}$ by P. N. Tan [5]

on the tree. In this example, consider the user-defined support value as 2, so item set $\{e\}$ is a frequent set because its support value is 3.

(iii) In second step, item $\{e\}$ is found as frequent, now algorithm tries to find frequent item sets with ending $\{d,\ e\}$, $\{c,\ e\}$, $\{b,\ e\}$ and $\{a,\ e\}$. Therefore, the prefix paths should be converted into a conditional FP-Tree. The conditional FP-Tree that is shown in Figure 2.4(b) is structurally similar to an FP-Tree. The only difference between these two trees are updated support values and pruned infrequent items. The conditional FP-Tree is used to find frequent item sets ending with a specific suffix.

(iv) In order to construct the conditional FP-Tree of item $\{e\}$, first, support values on the prefix paths must be updated because there are some transactions that do not contain item $\{e\}$. The rightmost path, for instance, shown in Figure 2.4(a), includes a transaction $\{b,\ c\}$ but not contain $\{e\}$. Therefore item counts on this path must be set to 1 to show the actual counts of transactions containing $\{b,\ c,\ e\}$. After that point, node $\{e\}$ can be removed from the prefix paths, because the support values on the prefix paths are updated to show only transactions that contain item $\{e\}$. After updating the support values, infrequent items can be removed from the tree. For instance, the support value of the node $\{b\}$ is equal to *1*, so it has lower support value than user-defined support, so it can be removed.

(v) The conditional FP-Tree for $\{e\}$ is used to find frequent item sets ending with $\{d, e\}$, $\{c, e\}$ and $\{a, e\}$. In order to find the frequent sets ending with $\{d, e\}$, the prefix paths of item $\{d\}$ are generated from the conditional FP-Tree for $\{e\}$ as seen in Figure 2.4(c). The support count of $\{d, e\}$ is found by following the links between item $\{d\}$, that is 2 in this example, so it is a frequent item set. After that, the algorithm constructs the conditional FP-Tree for $\{d, e\}$ that is shown in Figure 2.4(d). The conditional FP-Tree of $\{d, e\}$ contains only one item $\{a\}$ and its support is 2. The algorithm extracts the item set $\{a, d, e\}$ as frequent. The algorithm proceeds to solve all subproblems to find all frequent item sets. The prefix paths of item sets $\{c, e\}$ and $\{a, e\}$ can be seen in Figure Figure 2.4(e) and Figure 2.4(f) respectively.

(vi) The last process in FP-Growth algorithm is to find association rules from frequent item sets. For instance, algorithm finds $\{a, d, e\}$ as frequent item set. For this set, all possible rule combinations like $\{a \Rightarrow d\}$, $\{a \Rightarrow d, e\}$ are generated. The confidence values for all combinations are calculated and the rules that have greater confidence value than user-defined confidence value are mined as valid association rules.

2.2. Association Rule Mining on Data Streams

A data stream is a sequence of data that arrives in timely order. Unlike static databases, data streams arrive continuously with high speed and contains huge amount of data. As the number of applications that use data streams grows rapidly, there is an increasing need to perform association rule mining on stream data [28]. Some application areas of data mining on streams are sensor networks, manufacturing lines and web searches.

There are some challenging points on data stream mining topic. The most important one is because of rapid arrival rate of data, there is no multi-scan chance over the data. Also, due to the fast flow of data, rule mining algorithms should process the data as fast as possible. Besides these, another important point in stream mining is reasonable memory consumption, because it is not possible to store all stream data in memory.

There are three stream data processing models, Landmark, Damped and Sliding Windows [16].

The Landmark model mines all frequent item sets over the entire history of stream data from a specific time point called landmark to the present. Usually landmark point is set as a system start, so it means data mining is made over whole data. Therefore, this model is not suitable for applications where the most recent information of the data streams is valuable for the users.

The Damped model, mines frequent item sets in stream data in which each transaction has a weight and this weight decreases within time. That is, older transactions have less importance than new ones. This model can be used for applications in which old data has an effect on the mining results, but the effect decreases with time [28].

In the Sliding Windows model, mining process is made on sliding windows. In this approach, the part of data streams within the current window are stored and processed for mining. We can choose the sliding window size according to application domain and system resources. The mining result of the sliding window method totally depends on recent data in the range of the window.

The rule mining algorithms on data streams can be divided into two groups which are exact algorithms and approximate algorithms. In exact mining algorithms, the result sets contains all of the item sets which have greater or equal support value than the user defined support threshold. Approximate algorithms generate approximate result sets which are pruned with probabilistic guarantee. There are two possible approaches in approximate mining of frequent patterns with a probabilistic guarantee which are false positive oriented and false negative oriented. The first one includes some infrequent patterns in the result sets, whereas the second misses some frequent patterns [38].

2.3. Database as a Service Concept

Today, efficient data processing is a fundamental and vital issue for almost every scientific, academic or business organizations. However, this processing has been getting increasingly expensive and impractical if the database systems and problems are large and complicated [8]. Therefore, organizations usually want to outsource difficult works like data management and focus on their own business. This approach has brought the "database as a service" initiative to the industry.

In "Database as a service" paradigm requested database processes are provided by a service provider. For instance, operations like database administration, backup, migration and optimization is carried out by service provider [7]. However, database as a service leads new challenges and the most vital of them is providing data privacy. In this model, data owners share their valuable information with the service providers. Most corporations consider their data as a very valuable asset [8]. In the case of sharing data with the third parties, content or structure of the data should be hidden for security. Encryption is the most known and fundamental methodology in providing data security. In [37], authors have mentioned about the substitution cipher technique that is a well-known method for encryption of plain text.

2.4. Encryption & Security Issues

With the increasing usage of internet and network technologies, data exchange rates over the networks increase day by day. However, this wide usage also increase the undesirable attacks on network, so providing a secure exchange environment becomes a vital issue.

In order to protect the data from the attackers, the most used technique is the encryption. In order to encrypt the data, Advanced Encryption Standard (AES) can be used as an encryption standard. AES is a symmetric-key encryption standard adopted by the U.S. government. In symmetric-key encryption, same key, called secret key, is used for both encryption and decryption. Data encrypted with a secret key can be decrypted only with the same key. However, this shows the lack of this algorithm that is if the key is obtained in any type of attack, all data can be decrypted.

AES is based on a design principle known as a Substitution permutation network and it runs fast in both software and hardware. AES contains the following steps which are stated in [41]:

(i) *SubBytes:* Each byte in the array is updated using an 8-bit substitution box.
(ii) *ShiftRows:* Operates on the rows of the state; it cyclically shifts the bytes in each row by a certain offset.
(iii) *MixColumns:* The four bytes of each column of the state are combined using an invertible linear transformation.
(iv) *AddRoundKey:* The subkey is combined with the state.

The other encryption algorithm RSA, that is an algorithm for public-key cryptography, was first described by Rivest, Shamir and Adleman in [29]. In RSA algorithm, two keys that are public key and private key are generated. The public key can be known by everyone and is used for data encryption. The data that is encrypted with the public key can only be decrypted using the private key. The key generation phase of the RSA is summarized in [30] as follows:

(i) Choose two prime numbers p and q. For security purposes, the integers p and q should be chosen at random, and should be of similar to algorithm bit-length.
(ii) Compute $n = pq$ where n is used as the modulus for both the public and private keys.
(iii) Compute $\varphi(n) = (p - 1)(q - 1)$, where φ is Euler's totient function.
(iv) Choose an integer e such that $1 < e < \varphi(\mathrm{n})$ and $\gcd(e, \varphi(n)) = 1$ (e and $\varphi(\mathrm{n})$ are co-prime).

(v) Determine $d = e^{-1}(\text{mod } \varphi(n))$. ($d$ is the multiplicative inverse of e(mod φ(n))).

If the p and q integers are chosen similar to bit-length of the algorithm, the product of these two big prime numbers will be a very large number. Therefore, the finding the factors of the product number is slightly time consuming process and this makes the RSA so powerful and revolutionary compared to previous systems of encryption.

3. ARCHITECTURE OF THE PROPOSED SYSTEM

3.1. Problem Statement

Typical association rule mining algorithms, in the literature, especially find positively correlated association rules. However, mining negative correlated rules can provide us valuable information about the data like identifying products that conflict with each other or products that complement each other. Negative rule mining is more difficult task than positive rule mining, because of some differences between these two topics. In negative rule mining, the search space is much bigger than positive rule mining and this is the main reason of difficulty of this problem. Therefore, in the literature, lots of researches pointed out the negative association rule mining, but there are only few proposed algorithms about this topic [4, 11, 14, 15].

As mentioned in the previous section, today, data owners want to focus on their own business rather than managing their data. Therefore, data management issues are outsourced to third-party service providers and this concept is named as ”Database as a service”. ”Database as a service” concept meets lots of the expectations of data owners, but this concept raises an important security issue that is the data privacy protection. Data owners usually do not want to share their own private and valuable information with service providers [9]. Therefore, there have been various techniques in the literature [10, 21] to protect the data against the third-party players.

There are some challenges for association rule mining over data streams because of characteristics of streaming data. The main characteristics of data streams are continuous, unbounded and high speed. Traditional association rule mining algorithms, usually, make multiple scans over the data set to find frequent item sets. However, in data streams, there is a continuous data flow and there is no chance to make multiple scans on data sets. It means that traditional algorithms is not suitable for rule mining on streams. Due to the huge amount of data efficient memory space usage is required to mine streaming data. Also, because of high speed data flow, mining algorithms should process the data as fast as possible.

If summarize the previous paragraphs, the following problems are arise: First one is, can we implement an algorithm to mine positive and negative rules at the same time? Another problem is that can our proposed rule mining algorithm be run over the XML data stream? The last questions is that can we make all these processes in the "database as a service" concept with providing data privacy. Considering these there questions, our research question can be formed as: "Can a rule mining algorithm be implemented for both positive and negative rule mining over XML data stream with database a service concept to provide acceptable performance and reliable results?". In this research, we try to find an answer to this question.

3.2. Proposed Model

In order to find an answer to the research question introduced in the previous chapter, we have constructed PNRMXS algorithm. In this chapter, detailed information about the proposed model will be presented.

3.2.1. Model Assumptions

In our proposed model PNRMXS, we have the following assumptions:

(i) The average size of the each block of the data stream is constant (i.e each block contains n transactions).

(ii) Arriving items in a transaction are sorted.

(iii) In PNRMXS model, we only mine positive rules with form $X \Rightarrow Y$ and, for algorithm simplicity, negative rules with form $X \Rightarrow \neg Y$.

3.2.2. PNRMXS Algorithm

In this research, we propose PNRMXS model that is a single-scan algorithm for mining positive and negative rules on XML data stream in database as a service concept.

The whole processes in our algorithm, as shown in Figure 3.1, are run by two sides which are "data owner" and "service provider". The steps of PNRMXS algorithm are described in detail as follows with example data set in Figure 2.3.

(i) *Data Transformation:* Data transformation step is the first step of PNRMXS algorithm. In this step, a simple encryption method, one-to-one mapping, is used as proposed in [37]. Even if we work with a reliable service provider, we may want to hide the content of our data. For this purpose, we use a transformation map that contains the original item contents and the transformed item contents as key-value pairs. In PNRMXS implementation, the transformed items are generated with a random number generator. With the help of this mapping, the service provider only mines the association rules which contain transformed items and the original items can not be known directly. The most important point in this technique is that because of using one-to-one mapping the association rule mining algorithms can be applied with 100% accuracy. One-to-one item mapping for association rule mining gives us enough security, because the number of possible mappings is much larger, so guessing the mapping transformation is a hard process.

In order to provide more security, some other transformation techniques like adding fake items to original item set or one-to-many mapping can be used [37]. However, as you might guess, these processes are costly operations that has negative effect in stream mining and sometimes they may not give accurate mining results. It is very important to make a reasonable choice between security and performance issues. Therefore, taking into account on working streaming environment, we choose the one-to-one mapping technique to provide us necessary security.

Data transformation process is described in [37] as following: *m:* $I \rightarrow J$ is a substitution cipher, which maps the original set I of items to another set J, where

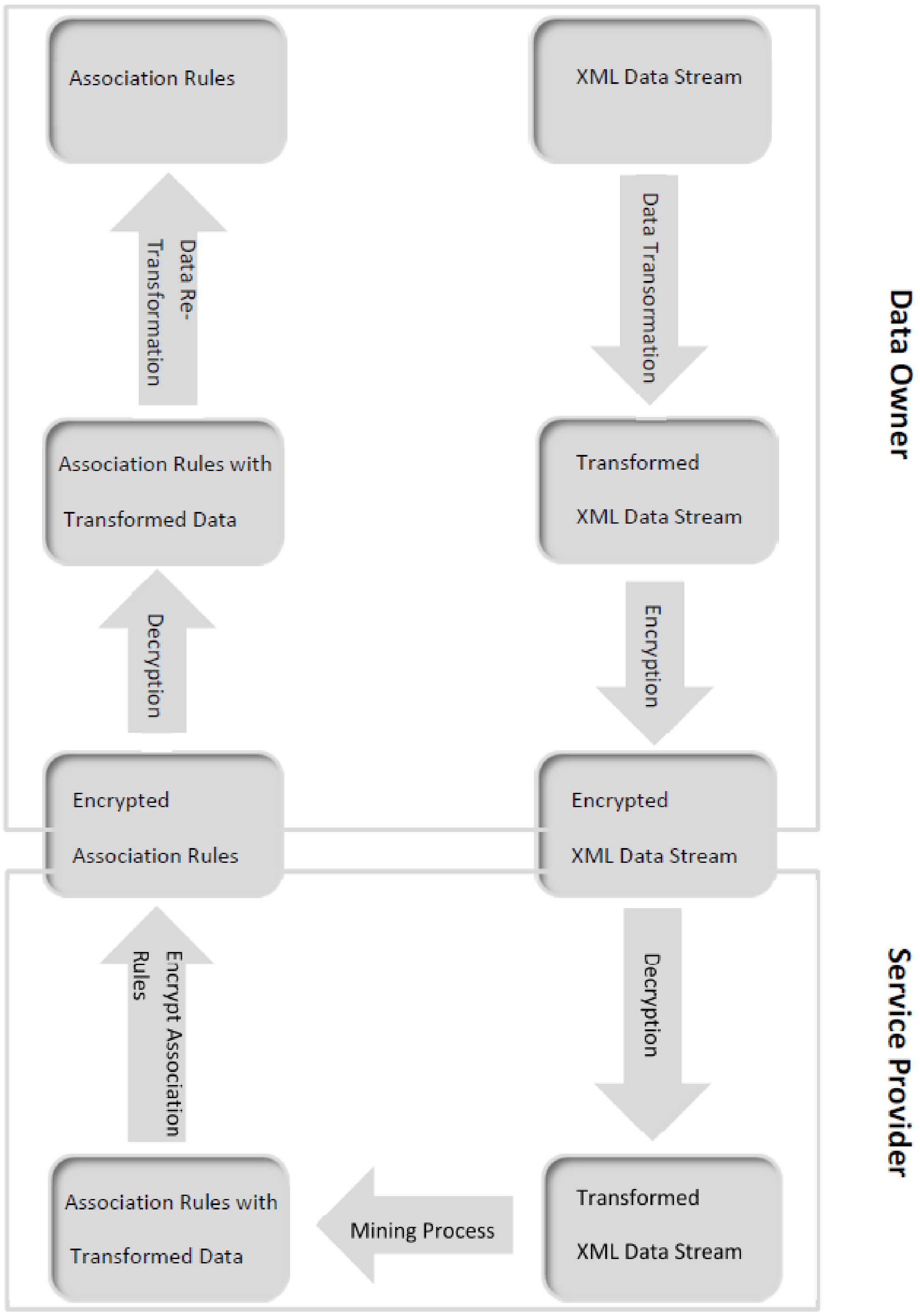
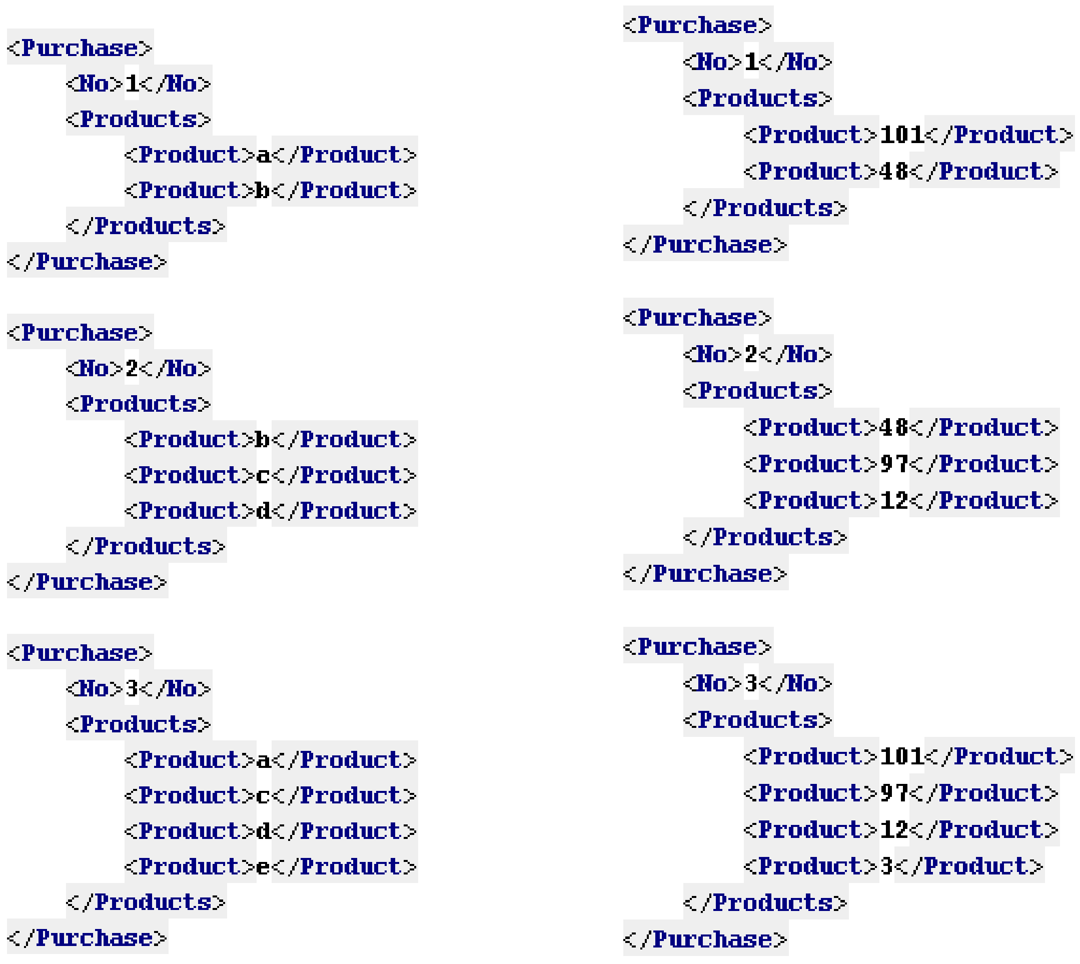

Association Rules
XML Data Stream
Data Re-Transformation
Data Transormation
Association Rules with Transformed Data
Transformed XML Data Stream
Decryption
Encryption
Encrypted Association Rules
Encrypted XML Data Stream
Encrypt Association Rules
Decryption
Association Rules with Transformed Data
Mining Process
Transformed XML Data Stream
Data Owner
Service Provider

Figure 3.1: Flowchart of the PNRMXS Algorithm

$|I| = |J|$. A transaction t_i is transformed to $M(t_i) = \{m(x) \mid x \in t_i\}$. For example, consider the transaction $t_i = \{a,\ b\}$ and assume that $m(a) = 101$ and $m(b) = 48$. As a result, t_i is transformed to $M(t_i) = \{101,\ 48\}$. In Figure 3.2, you can see an example content of the XML data with original and transformed items.



```
<Purchase>
    <No>1</No>
    <Products>
        <Product>a</Product>
        <Product>b</Product>
    </Products>
</Purchase>

<Purchase>
    <No>2</No>
    <Products>
        <Product>b</Product>
        <Product>c</Product>
        <Product>d</Product>
    </Products>
</Purchase>

<Purchase>
    <No>3</No>
    <Products>
        <Product>a</Product>
        <Product>c</Product>
        <Product>d</Product>
        <Product>e</Product>
    </Products>
</Purchase>
```

(a) Original XML

```
<Purchase>
    <No>1</No>
    <Products>
        <Product>101</Product>
        <Product>48</Product>
    </Products>
</Purchase>

<Purchase>
    <No>2</No>
    <Products>
        <Product>48</Product>
        <Product>97</Product>
        <Product>12</Product>
    </Products>
</Purchase>

<Purchase>
    <No>3</No>
    <Products>
        <Product>101</Product>
        <Product>97</Product>
        <Product>12</Product>
        <Product>3</Product>
    </Products>
</Purchase>
```

(b) Transformed XML

Figure 3.2: Example XML Data

(ii) *Stream Encryption:* In encryption step; AES (Advanced Encryption Standard) is used to encrypt the transformed XML data. AES is a symmetric-key encryption technique that means encryption and decryption processes should be made with same key. AES has a fixed block size of 128 bits and a key size of 128, 192, or 256 bits, here we choose 128 bits key size, because 128 bits is currently thought, by many observers, to be sufficient and the U.S. government requires 192 or 256-bit

AES keys for highly sensitive data [36].

In our algorithm, first, AES key is generated and stored in a binary file and after that the transformed XML data is encrypted with this key. Now, our XML data is secure. However, the service provider should now the AES key for decryption, but how can we transmit the AES key file in a secure way to the service provider? The problem here is that in order to distribute encryption keys between the data owner and the service provider, another secure transmission environment should be provided. All classical encryption methods suffer from this key distribution problem.

We will use another encryption approach to solve the problem explained above. In order to distribute the AES key file securely between the parties, RSA encryption algorithm will be used. As explained in Section 2.4, RSA is an asymmetric encryption technique that is encryption is made by a public key, but decryption can be made only a private key. The public key can be distributed even in unsecured ways between the parties, but the private key should be kept securely. In this step, the service provider generate a public-private key pair and transmit the public key to the data owner. The data owner will encrypt the AES key file with the RSA public key. From now on, our XML data and AES key file is secure and can be distributed safely. At that point, this question can be asked. Why do not we use RSA to encrypt and decrypt the whole data stream? The answer is that RSA encryption is slower than AES encryption in large data, so in order to decrease the execution time of algorithm, we use RSA for only AES key file encryption. Also, in our implementation, we will use RSA key length as 1024 bits.

(iii) *Stream Decryption:* The third process is the reserve of the "*Stream Encryption*" step. This task is made by service provider. After the data owner transmits the encrypted XML data stream and AES key file to the provider, first AES key file is decrypted with private key which is hold by provider. Then, XML data stream is decrypted with AES key. The graphical notation of "*Stream Encryption*" and "*Stream Decryption*" processes can be seen in Figure 3.3.

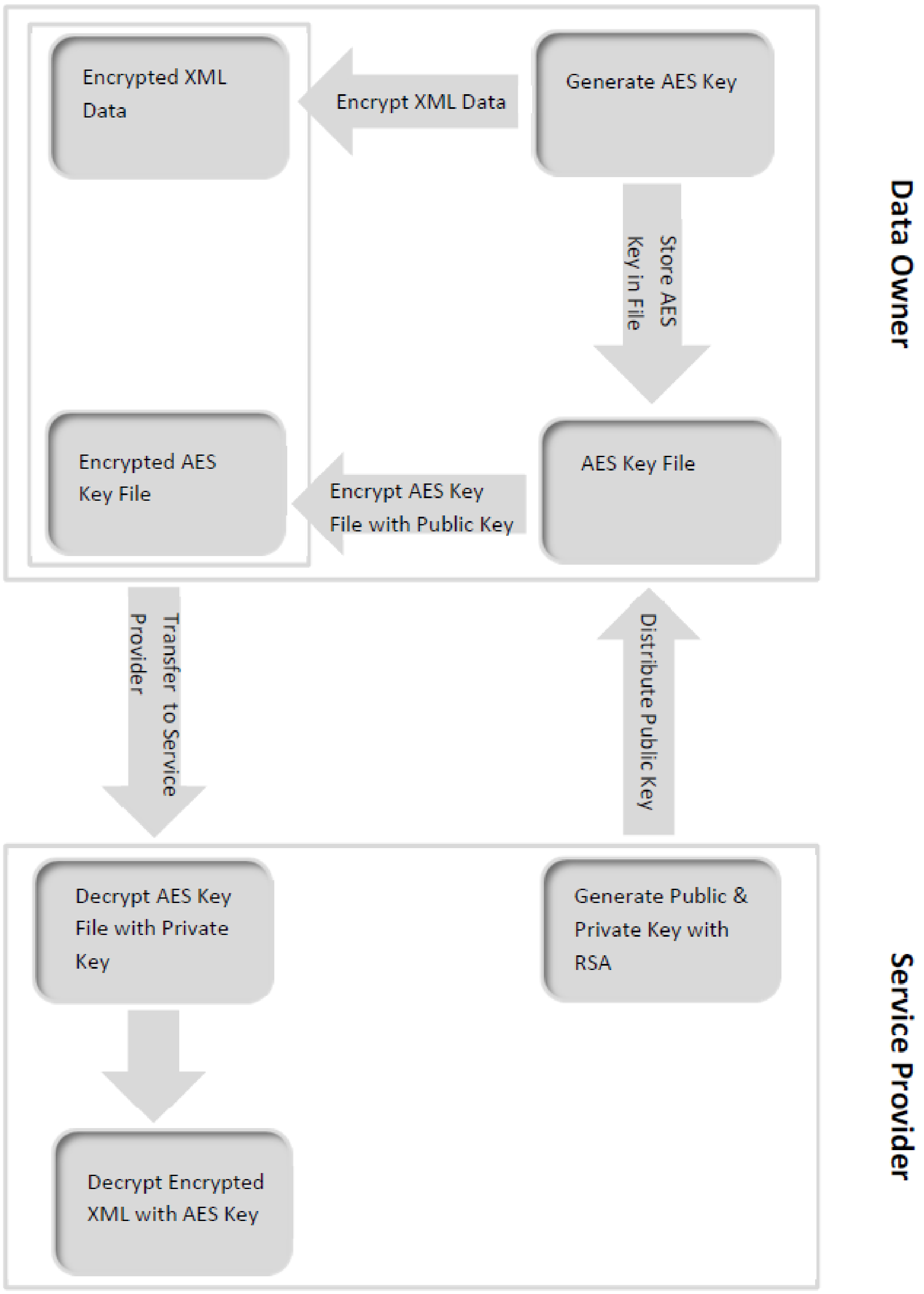

Data Owner
Encrypted XML Data
Encrypt XML Data
Generate AES Key
Store AES Key in File
Encrypted AES Key File
Encrypt AES Key File with Public Key
AES Key File
Transfer to Service Provider
Distribute Public Key
Service Provider
Decrypt AES Key File with Private Key
Generate Public & Private Key with RSA
Decrypt Encrypted XML with AES Key

Figure 3.3: Flowchart of Stream Encryption & Decryption Processes

(iv) *Mining Process:* This is the main step in our PNRMXS algorithm. The proposed model uses two models as a basis which are landmark windows model mentioned in [16] and FP-Growth in [3].

The landmark windows model is used as a stream data processing model in our proposed system. In this approach, data mining process is run over the data between a specific window identifier called landmark and the present window identifier. Landmark usually indicates the system start time and with this model the rule mining is made over the entire data set.

The mining procedure is developed by using FP-Growth algorithm approach. However, as stated before, the original FP-Growth algorithm is not suitable both for rule mining on data streams and negative rule mining. Therefore, we should make some adaptations on the algorithm.

The first change in the algorithm, because of the continuous characteristics of data stream, is that the data set should be scanned only once. The data stream, in this model, is processed block by block. As stated in our assumptions, each block contains fixed number of transactions. The aim of PNRMXS algorithm is mining positive and negative association rules at a time. In order to provide this capability, items in the transactions are not eliminated according their frequencies and all items are mapped to the FP-Tree. At first glance, it may seem to be losing the advantage of the FP-Growth algorithm about memory space saving. However, this is a required change for negative association rule mining, because without all items negative rules can not be mined correctly. For example, another negative rule mining approach is proposed in [14]. They mine the negative rules based on the existing positive rules. This approach is decrease the search space for mining negative rules efficiently, but loose lots of negative relations in the data. Finding valid and enough number of negative associations is as important as saving memory space.

The finding frequent item set phase in our algorithm is the similar to the original

FP-Growth algorithm as explained in Section 2.1. As stated before, in standard support-confidence framework there is only one support and one confidence threshold value. However, these parameters do not provide sufficient pruning capability for negative rule mining, so in PNRMXS algorithm, we also add new extra support and confidence thresholds. In our approach, we do not use any threshold value to eliminate infrequent items. Support threshold are only used to prune frequent sets. Confidence values, in PNRMXS algorithm, are used in only rule generation phase. In the next paragraph, we will give our threshold definitions.

There are five threshold values which are *"MS"*, *"MSP"*, *"MCP"*, *"MSN"* and *"MCN"* in our model. We define the all thresholds, except "*MS*", as the same manner with proposed model "*PNAR*" in [17]. The "*PNAR*" model is an Apriori-based algorithm. It first separates the items as frequent and infrequent with minimum support threshold. After that, generate positive rules from frequent items with support and confidence threshold and negative rules from infrequent items with different support and different confidence threshold. In PNRMXS, all items in the data set are put in to our FP-Tree without any elimination. In finding frequent set phase, because of containing all items, there is huge amount of frequent sets. Even if, in negative rule mining case, most of these frequent sets can be invaluable for data owner. Therefore, in order to make extra pruning and gain extra performance, we also adapt "*MS*" threshold in our algorithm. If an item set has an support value greater than "*MS*", it can be a rule candidate. Another adapted pruning strategy in our model is "*Correlation Coefficient*" value as used in [13, 17]. However, in [13], authors define a threshold value for correlation coefficient value over Apriori-based algorithm and until finding positive or negative correlated items, this threshold value is lowered. Lowering the correlation threshold causes additional scans that is not suitable for stream mining. Therefore, in our model we do not define threshold value for correlation coefficient value. In PNRMXS, we use the same correlation coefficient approach with [17], but we use FP-growth algorithm as basis, on the other hand, they use Apriori algorithm as basis. As defined in [26], "*Correlation Coefficient*" measures the degree of the linear relationship between a pair of random variables. In order to understand in detail, we use the definitions given in [13, 26].

Table 3.1: 2x2 Contingency Table for Binary Variables by M. L. Antonie [13]

	Y	$\overline{Y}$	Σrow
X	f_{11}	f_{10}	f_{1+}
$\overline{X}$	f_{01}	f_{00}	f_{0+}
$\Sigma column$	f_{+1}	f_{+0}	N

The correlation coefficient value between the two variables X and Y can be calculated with the formula in Equation 3.1:

$$\rho_{XY} = \frac{Cov(X,Y)}{\sigma_X \sigma_Y} \tag{3.1}$$

The correlation coefficient is defined as the covariance of the two variables that is *Cov(X, Y)* divided by the product of their standard deviations that is σ. The correlation value ranges from -1 and +1. If ρ_{XY} is equal to 0, it indicates that these two variables are independent. The positive ρ_{XY} value means the variables are positively correlated and negative value means the variables are negatively correlated. If the coefficient value is close to either -1 or +1, there is a strong or perfect correlation between the variables.

The cells of the Table 3.1 represent the possible combinations of variables X and Y and give the frequency associated with each combination. N is the size of the data set. With using the contingency table, Pearson defined the correlation coefficient as in the Equation 3.2.

$$\phi = \frac{f_{11}f_{00} - f_{10}f_{01}}{\sqrt{f_{1+}f_{0+}f_{+1}f_{+0}}} \tag{3.2}$$

"*Correlation Coefficient*" parameter is used for frequent item sets separation in our model as used in other models. If item set has a "*Correlation Coefficient*" value greater than 0 it has positive correlation and if it has a value less than 0 it has negative correlation. After finding all frequent sets with using "*MS*" threshold and after separating them as positive or negative, rule generation process is started. The positive correlated frequent sets is used to mine positive association rules with using "*MSP*" and "*MCP*" thresholds. If a frequent item set has negative correlation, it may be a possible negative rule candidate. If the item set has negative correlation, first, the possible rules with structure "$X \Rightarrow Y$" are generated from this item set. Then, the consequent part of the rule, that is "Y" in this notation, is negated. Lastly, the support and confidence values of the rule candidate are compared with "*MSN*" and "*MCN*" thresholds to decide if it is a valid negative rule or not. The pseudo-code of PNRMXS algorithm can be seen in Figure 3.4 and 3.5.

```
Procedure PNRMXS(DS, MS, MSP, MCP, MSN, MCN)
1: Define and clear the root node of FP-Tree; root;
2: for all Block $B_i \in DS$ do
3:     for all Transaction $T_i \in B_i$ do
4:         Call CreateTree($T_i$, root);
5:     end for
6: end for
7: for all item $a_i \in I$ do
8:     Call PNRMXS_Growth(root, $a_i$, MS, MSP, MCP, MSN, MCN);
9: end for
```

Figure 3.4: PNRMXS Algorithm Pseudo-Code

```
Procedure PNRMXS_Growth(root, a, MS, MSP, MCP, MSN, MCN)
1: if root contains single path P then
2:   for all combination(c) of the nodes $\in P$ do
3:     Generate pattern $p = c \cup a$
4:     if $p.support \geq MS$ then
5:       if $corr(p) > 0$ then
6:         Generate Rule pr from p as $(X \Rightarrow Y)$
7:         if $pr.support \geq MSP$ and $pr.confidence \geq MCP$ then
8:           Call Output(pr);
9:         end if
10:      else if $corr(p) < 0$ then
11:        Generate Rule nr from p as $(X \Rightarrow \neg Y)$
12:        if $nr.support \geq MSN$ and $nr.confidence \geq MCN$ then
13:          Call Output(nr);
14:        end if
15:      end if
16:    end if
17:  end for
18: else
19:  for all $b_i \in root$ do
20:    Generate pattern $p = b_i \cup a$
21:    if $p.support \geq MS$ then
22:      if $corr(p) > 0$ then
23:        Generate Rule pr from p as $(X \Rightarrow Y)$
24:        if $pr.support \geq MSP$ and $pr.confidence \geq MCP$ then
25:          Call Output(pr);
26:        end if
27:      else if $corr(p) < 0$ then
28:        Generate Rule nr from p as $(X \Rightarrow \neg Y)$
29:        if $nr.support \geq MSN$ and $nr.confidence \geq MCN$ then
```

```
30:           Call Output(nr);
31:         end if
32:       end if
33:     end if
34:     Construct p's conditional pattern base and conditional FP-Tree $CTree_p$
35:     if $CTree_p \neq \emptyset$ then
36:       Call PNRMXS_Growth ($CTree_p$, p, MS, MSP, MCP, MSN, MCN)
37:     end if
38:   end for
39: end if
```

Figure 3.5: PNRMXS Algorithm Pseudo-Code

(v) *Encrypt Association Rules:* After positive and negative rules are mined by the service provider, they are serialized to a binary file. However, as mentioned in previous encryption step, in order to provide security, this file also should be encrypted. We are using the same methodology as mentioned in "*Stream Encryption*" step, but in the reverse way. First, data owner generates a public-private key pair with RSA algorithm and transmit the public key to the provider. Service provider encrypts the binary file that contains positive and negative association rules with AES key and also encrypts the AES key file with public key sending from data owner. Lastly, encrypted key file and binary file is sent to data owner.

(vi) *Decrypt Association Rules:* The responsibility of this process is upon data owner. Data owner, first, decrypts the AES key file with its own private key and then decrypts the encrypted binary file is decrypted with this AES key.

(vii) *Data Re-Transformation:* The last step in PNRMXS is data re-transformation. In this step, we have the positive and negative association rules with transformed items. For example, we find $\{48 \Rightarrow 101\}$ positive association rule, we re-transform the items with using mapping table that is generated in "*Data Transformation*" step and finally get the $\{b \Rightarrow a\}$ rule.

4. EXPERIMENTAL RESULTS

In this section, the performance and scalability evaluation of the PNRMXS algorithm will be presented. Throughout this section, experiments are designed to provide valid and reliable results. We, in the following sub sections, will identify all the steps of the experiments.

4.1. Experimental Data Sets

In our experiments, we will only use generated synthetic data sets, because of some reasons. First of all, synthetic data sets are generated to meet specific needs or certain conditions that may not be found in the real data sets [20]. Another reason is that, with synthetic data sets, we can analyze the scalability of our algorithm with ease, because we can generate data sets with different size. Finding real data sets to test the mining algorithms is difficult issue, because the real data sets can contain confidential information and owners do not want to share them. With using synthetic data set, we can also handle this problem.

4.1.1. IBM Synthetic Data Set Generator

In the experiments that are performed throughout this research, we will use IBM synthetic data generator tool that is proposed by [2]. This tool is widely used in literature to generate experimental data sets.

However, IBM data generator only generates data sets with flat file structure. In order to convert this data to XML structured data, we implement a flat file to XML converter in Java.

We will use the Table 4.1 as a reference table throughout this section for parameters descriptions. For the whole experiments in this sections, the data sets that are shown in Table 4.2 will be used. We use lots of data sets to get reliable results about our proposed model performance and scalability.

Table 4.1: Descriptions of Parameters

Parameter Name	Description
D	Number of transactions in data stream
I	Average length of maximal pattern
T	Average length of transaction
N	Number of distinct items
K	Thousands
MS	Minimum support threshold
MSP	Minimum support threshold for positive rules
MCP	Minimum confidence threshold for negative rules
MSN	Minimum support threshold for negative rules
MCN	Minimum confidence threshold for negative rules

Table 4.2: Generated Test Data Sets

Data Set Name	**File Size(MB)**	***T***	***N***	***I***	***D***
T5N1000I4D100K	16.5	5	1000	4	100,000
T5N2000I4D100K	16.8	5	2000	4	100,000
T5N4000I4D100K	16.9	5	4000	4	100,000
T10N1000I4D100K	28.8	10	1000	4	100,000
T10N2000I4D100K	29.4	10	2000	4	100,000
T10N4000I4D100K	29.7	10	4000	4	100,000
T10N1000I4D200K	57.8	10	1000	4	200,000
T10N2000I4D200K	58.9	10	2000	4	200,000
T10N4000I4D200K	59.5	10	4000	4	200,000

As seen in Table 4.2, the differences between the data sets are that they have different number of items, different average length of transactions and different number of transactions. The data are broken into blocks of size 20K to simulate the continuous characteristics of streaming data throughout the experiments unless otherwise stated.

4.2. Experimental Environment

The PNRMXS algorithm is developed by Java programming language. We use the proposed code in [43] as a base and we made our adaptation on this implementation. The test workstation has Intel Xeon 2.27 Ghz processor and 3 GB RAM. The operating system is Windows 2003 with Service Pack 2 and installed Java version is 1.6.0_12. Also, BIRT (Business Intelligence and Reporting Tools) tool is used to prepare the experiments' graphical notation.

4.3. Experimental Results

The every steps in PNRMXS algorithm are implemented, but we focus on the service provider steps especially, so the experimental results will be related with these steps. For experiment result, we would like to give the below information also:

- All given results about execution times are in milliseconds,
- All given results about memory usages are in Megabytes,
- All test are made on our test workstation without any extra load and with only one log on user.

4.3.1. FP-Tree Node Counts

As stated before, in FP-Growth algorithm, data set is mapped to an FP-Tree. Before the mapping process, the data set is pruned from infrequent items and this causes the extra scan over the data. Due to the structure of the FP-Tree, there are lots of overlapped paths. Therefore, this means that the node(item) count in the tree is lower than the original item count of data set. In our proposed model, the pruning process is eliminated in order to adapt for stream mining and find negative association rules. We put all items to the tree, so we sacrifice some memory space. However, we still have lower node(item) count than the original data set, because of overlapping tree paths. In Figure 4.1, we will show how the FP-Tree node counts change with different data sets.

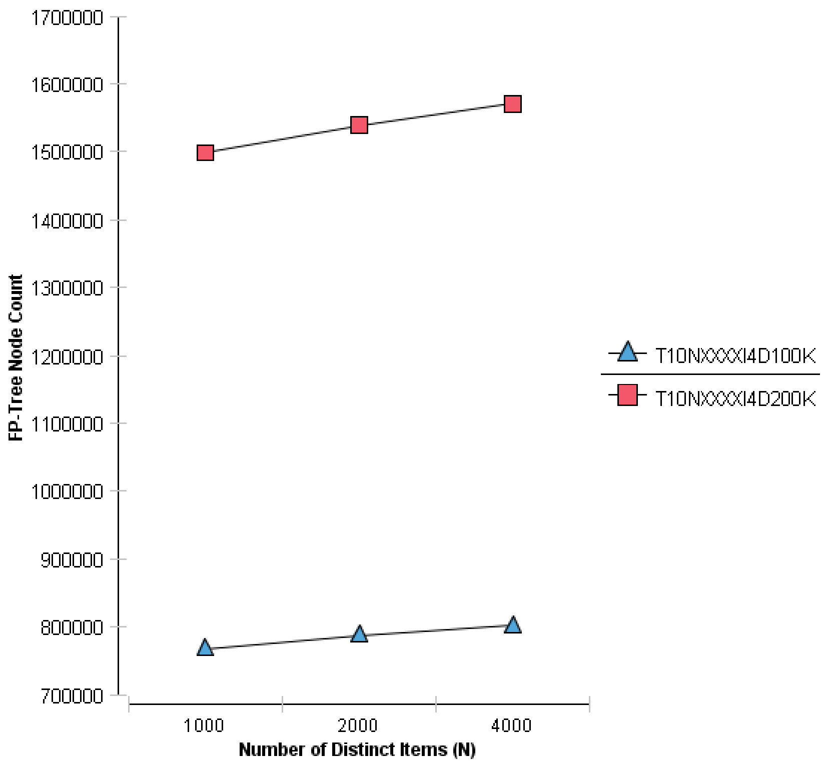


Figure 4.1: FP-Tree Node Counts

In this experiment, six data sets are compared. In "triangle" and "square" labeled graphs, test data sets have approximately 1,000,000 and 2,000,000 items respectively. As expected, because of the overlapping paths, node counts range from 700,000 to 800,000 and 1,500,000 to 1,600,000. Here, we want to address another issue is that if we increase the number of distinct items, the node counts are increased also. Since, increase in number of distinct items causes decrease in overlapping paths. This means that if we have a dense data set, out tree structure will be more compact. For example, the data sets T10N1000I4D100K and T10N2000I4D100K have approximately same item counts, but, the second one is more sparse, so it has more nodes. This explanation can be seen in detail in Table 4.3.

4.3.2. FP-Tree Generation Time Comparison

FP-Tree generation time is another important comparison criteria. In PNRMXS, data stream blocks first read and stored in main memory. After reading all blocks in

Table 4.3: FP-Tree Node Counts in Detail

Data Set Name	**Total Item Counts**	**Tree Node Counts**
T10N1000I4D100K	$\sim 1,000,000$	768,506
T10N2000I4D100K	$\sim 1,000,000$	788,279
T10N4000I4D100K	$\sim 1,000,000$	802,154
T10N1000I4D200K	$\sim 2,000,000$	1,498,780
T10N2000I4D200K	$\sim 2,000,000$	1,539,070
T10N4000I4D200K	$\sim 2,000,000$	1,570,413

the stream window, the FP-Tree is generated. Therefore, we can clearly say that PN-RMXS is a memory-based algorithm. Unlike original FP-Growth algorithm, we do not make any pruning process at the beginning that is thought as initial pruning threshold value is zero. Therefore, in PNRMXS algorithm, there are more nodes in the FP-Tree than the original FP-Growth and this leads increase in FP-Tree generation time.

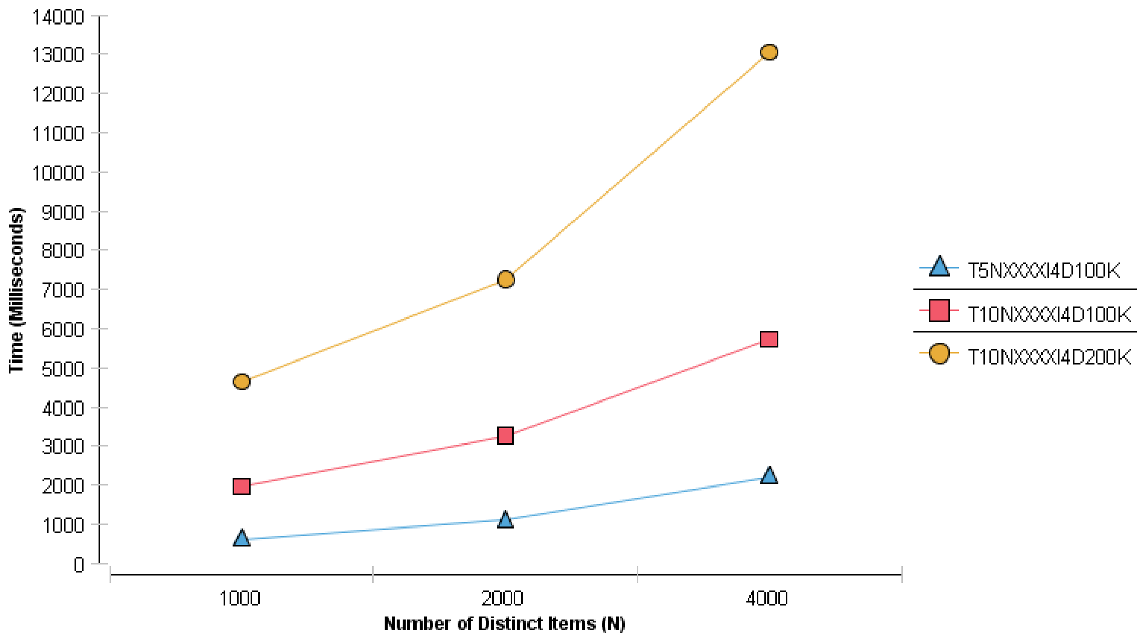


Figure 4.2: FP-Tree Generation Time Comparison

In Figure 4.2, we can see the FP-Tree generation time change depending on number of distinct items. If number of distinct items is increased, data distribution in data set changes and it causes less common items in transactions. Less common items means that there will be more paths in the FP-Tree and it causes more tree traversing. While number of distinct items is increasing, total FP-Tree generation time is also increasing, because number of common items is decreasing. In Table 4.4, the effect of number of distinct items parameter can be seen in detail. Between the data sets T10N4000I4D100K and T10N1000I4D200K, there is a big difference in node counts. However, tree generation takes less time with using second data set than the first one. This shows that the content of the data set has a determinative effect on algorithm time.

Table 4.4: FP-Tree Generation Times

Data Set Name	**Tree Node Counts**	**Tree Generation Time**
T5N1000I4D100K	248,540	625
T5N2000I4D100K	255,065	1109
T5N4000I4D100K	259,227	2219
T10N1000I4D100K	768,506	1969
T10N2000I4D100K	788,279	3266
T10N4000I4D100K	802,154	5724
T10N1000I4D200K	1,498,780	4658
T10N2000I4D200K	1,539,070	7251
T10N4000I4D200K	1,570,413	13051

4.3.3. Total Execution Time Comparison

In this experiment, we will show the total execution time of our proposed PNR-MXS algorithm with different parameter settings and different data sets. These results contain the only the service provider processes.

Table 4.5: Execution Times on T5N1000I4D100K data set

Data Set Name	**MS**	**MSP**	**MCP**	**MSN**	**MCN**	**Time**
T5N1000I4D100K	0.1%	0.6%	5%	0.5%	5%	4004
T5N1000I4D100K	0.1%	0.6%	1%	0.5%	1%	3942
T5N1000I4D100K	0.1%	0.5%	0.5%	0.4%	0.5%	4048
T5N1000I4D100K	0.1%	0.4%	0.5%	0.3%	0.5%	3921

The experiment results can be seen in Table 4.5 and 4.6. We use T5N1000I4D100K and T10N1000I4D100K data sets with different parameter settings, but we keep " *MS*"

Table 4.6: Execution Times on T10N1000I4D100K data set

Data Set Name	**MS**	**MSP**	**MCP**	**MSN**	**MCN**	**Time**
T10N1000I4D100K	0.1%	0.6%	5%	0.5%	5%	10764
T10N1000I4D100K	0.1%	0.6%	1%	0.5%	1%	10885
T10N1000I4D100K	0.1%	0.5%	0.5%	0.4%	0.5%	11137
T10N1000I4D100K	0.1%	0.4%	0.5%	0.3%	0.5%	10921

parameter constant. As previously mentioned, in PNRMXS algorithm, first, all frequent item sets are found by using "*MS*" threshold and after that they are pruned with correlation, support and confidence parameters to find positive and negative rules. If we look at execution times of two experiment sets, the results are almost same, even though parameters have different values, except "*MS*". Therefore, "*MS*" parameter can be commented as the major factor on algorithm execution time.

In order to support our argument, you can see the another experiment in Table 4.7 with different "*MS*" values. As seen in this test, in contrast to previous experiments, all parameter are kept constant except "*MS*". As expected, if the "*MS*" threshold increase, the total execution time decrease. Also, with the help of these results , we can easily say that our algorithm performance is stabile with different parameter settings.

Table 4.7: Execution Times on T10N1000I4D100K data set

Data Set Name	**MS**	**MSP**	**MCP**	**MSN**	**MCN**	**Time**
T10N1000I4D100K	0.07%	0.7%	1%	0.5%	1%	11837
T10N1000I4D100K	0.08%	0.7%	1%	0.5%	1%	11017
T10N1000I4D100K	0.09%	0.7%	1%	0.5%	1%	10875
T10N1000I4D100K	0.1%	0.7%	1%	0.5%	1%	10563

In previous tables, the total execution times of the different experiments are shown. However, which parts of PNRMXS algorithm take how much time to execute. It is an important detail, because it can give us which part of the algorithm can be improved in the further studies. In order to point out the execution time details, we test our algorithm with T10N1000I4D200K data set and with different threshold values.

At that point, we think that explication of these steps will be useful. As can be seen in Table 4.8, the most part of the execution time is spent on the ”XML Processes” step. This step includes the XML decryption and XML parsing operations. These processes are independent from all parameter settings and used data structures, hereby execution time values are close in all tests. In ”FP-Tree Generation” phase, as stated before, all items in the data set is mapped to the FP-Tree. Because of using same data set in all test, we have same number or tree nodes and so we get similar execution times. The reason of these small differences may be due to the instant computer loads. The ”Rule Mining” step consists of ”Rule Mining” and ”Rule Generation” processes. In this step, our pruning thresholds has a critical role. If we examine the table, increasing in threshold values causes decreasing in execution time, because of decreasing in number of frequent item sets. The most of the running time of this step is passing in ”Rule Mining” process that finds the all frequent item sets. The process of the ”Rule Generation” is more simple and quick process. It is quick because there is no extra operation about the data set and it is simple because there are only threshold value calculation and comparison processes. In order to verify this statement, we made another experiment and put the results in Table 4.9. As seen in table, it takes so little time. The last step is ”Rule Output” operation. In this step, mined rules are serialized to a file, before sending back to data owner. This takes almost no time as you can see in the results.

Finally, we would like to say that, these execution times show the total processing time for one stream window not one stream block, so these results are acceptable.

4.3.4. Execution Time Comparison of PNRMXS & Moment Algorithm

The aim of this experiment is to compare our PNRMXS algorithm with Moment algorithm that is proposed by Chi et al. [27]. Moment is developed in order to mine frequent closed item sets over data streams with sliding window model. The authors proposed an in data structure that is called CET (Closed Enumeration Tree) to store the data. Moment is an exact rule mining method that is they do not prune infrequent items at the stream processing step like we did. Also, they only use a minimum support

Table 4.8: Execution Time Details

Data Set Name	**XML Processes**	**FP-Tree Generation**	**Rule Mining**	**Rule Output**	**Time**
MS = 0.1% MSP = 0.5% MCP = 1% MSN= 0.5% MCN=1%					
T10N1000I4D200K	9243	4031	5742	~ 0	19016
MS = 0.07% MSP = 0.4% MCP = 5% MSN= 0.4% MCN=5%					
T10N1000I4D200K	10016	4328	7312	~ 0	21656
MS = 0.2% MSP = 0.6% MCP = 10% MSN= 0.5% MCN=10%					
T10N1000I4D200K	9562	3922	3328	~ 0	16812
MS = 0.2% MSP = 1% MCP = 10% MSN= 1% MCN=10%					
T10N1000I4D200K	9250	4000	3125	~ 0	16375

Table 4.9: Rule Generation Times on T10N1000I4D100K - T10N2000I4D100K

Data Set Name	**MS**	**MSP**	**MCP**	**MSN**	**MCN**	**Rule Count (PR & NR)**	**Time**
T10N1000I4D100K	0.09%	0.5%	1%	0.4%	1%	2 & 1792	75
T10N1000I4D100K	0.09%	0.4%	1%	0.3%	1%	34 & 1792	81
T10N2000I4D100K	0.09%	0.5%	1%	0.4%	1%	0 & 12	62
T10N2000I4D100K	0.09%	0.4%	1%	0.3%	1%	2 & 12	63

threshold to find closed frequent item sets and they do not make any rule generation process.

To the best of our knowledge, our proposed model have not previously defined by any other researcher, so there is no candidate model for one-to-one comparison. For instance, Moment algorithm only finds closed frequent item sets, but we find all frequent item sets. Also, while we make positive and negative rule generation process, in Moment algorithm, there is no rule generation operation. Because of these reasons, we can not compare these two models one-to-one. However, in order to give an idea for our model performance, we will compare the finding frequent sets process of our model with Moment algorithm.

For the comparison, we use the provided Moment algorithm code in [33]. This code was written in C++ language, we use "*mingw gcc*" linux compiler to create an executable file to be able to work in our test environment. In this test, we use real world dense data set *mushroom* which is provided in [44]. The test result is shown in Figure 4.3.

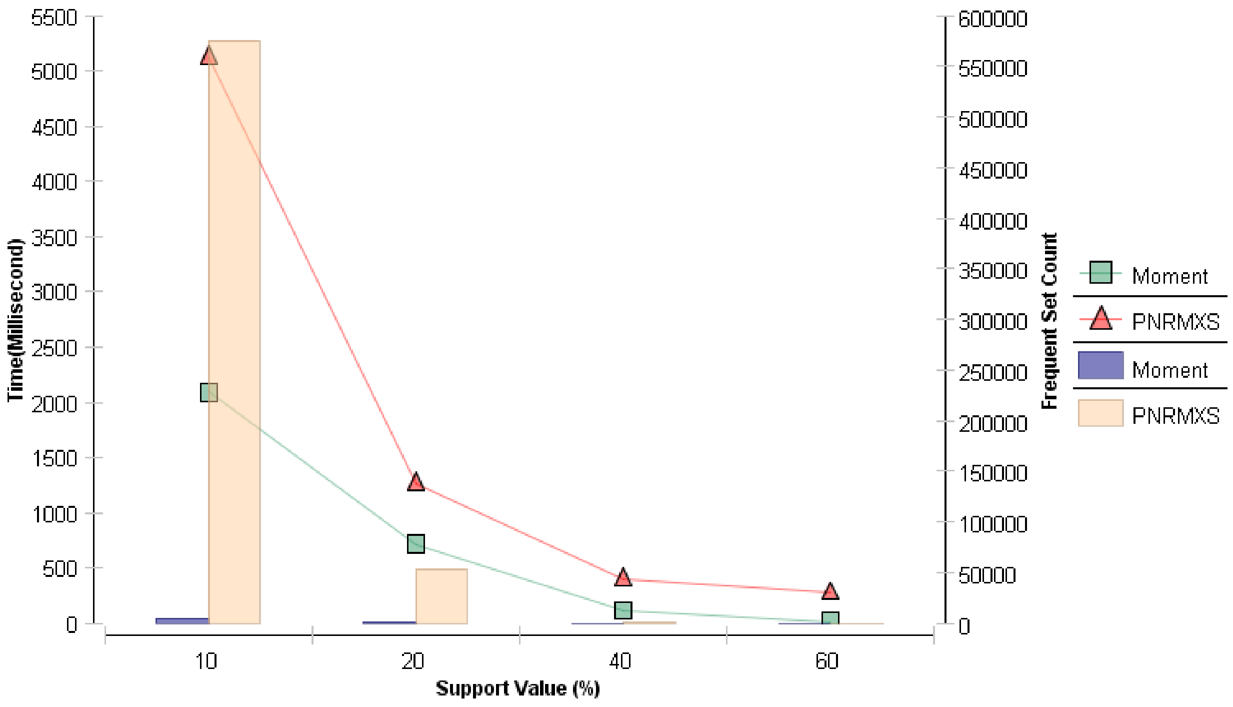


Figure 4.3: Moment & PNRMXS Comparison

At the first sight, if the execution times of the models are compared, Moment algorithm gives better results. However, at that point, the big difference between generated frequent sets count are taken into account. The main reason of this case is that Moment algorithm generates only closed frequent item sets, but PNRMXS algorithm finds all frequent item sets. With low support values, the difference goes up to approximately 100 times. To the contrary, the differences between the execution times is reasonable. Therefore, with these results, we can assert that our proposed model is applicable.

4.3.5. Memory Consumption Comparison

In this section, we will give the total size of memory consumption of PNRMXS algorithm with different data sets and threshold parameters.

Table 4.10: Memory Consumption Comparison

Data Set Name	**MS**	**MSP**	**MCP**	**MSN**	**MCN**	**Rule Count**	**Total Memory**
T5N1000I4D100K	0.1%	0.6%	1%	0.5%	1%	2	36.18
T10N1000I4D100K	0.1%	0.6%	1%	0.5%	1%	1202	42.84
T10N1000I4D200K	0.1%	0.6%	1%	0.5%	1%	1244	103.43

In the first experiment, threshold values are kept constant and different data sets are used. The data sets, in Table 4.10, are ordered according to the FP-Tree node counts. For detail information about the node counts, you can refer to Table 4.3. Between the first two data sets, there is a big difference in FP-Tree node counts and mined rule counts. As expected, increase in node counts and rule counts cause to increase in memory consumption. However, with the third data set, our algorithm give high memory usage than we expected. This may be due to the characteristics of data set that we generated.

In the second experiment, we use only one data set that is chosen in sparse ones and make the tests with different threshold values. As explained before, in PNRMXS algorithm, there is no pruning process in FP-Tree generation phase, so even if we use

Table 4.11: Memory Consumption for T10N4000I4D100K data set

Data Set Name	**MS**	**MSP**	**MCP**	**MSN**	**MCN**	**Rule Count**	**Total Memory**
T10N4000I4D100K	0.08%	0.2%	1%	0.2%	1%	36	130.32
T10N4000I4D100K	0.09%	0.3%	1%	0.3%	1%	18	113.29
T10N4000I4D100K	0.1%	0.4%	1%	0.4%	1%	0	111.40

different threshold values our tree has same node count. Therefore, here, we expect that the memory consumption should be similar in all tests. As seen in Table 4.11, PNRMXS algorithm gives us the stabile results in memory consumption. Also, it is obvious that the difference in rule counts brings a difference in memory usages.

4.3.6. Effect of Block Size on Execution Time

This test is performed to show the effect of block size of stream on execution time of PNRMXS algorithm. The T10N4000I4D100K data set is used and all parameters are kept constant while changing the block size.

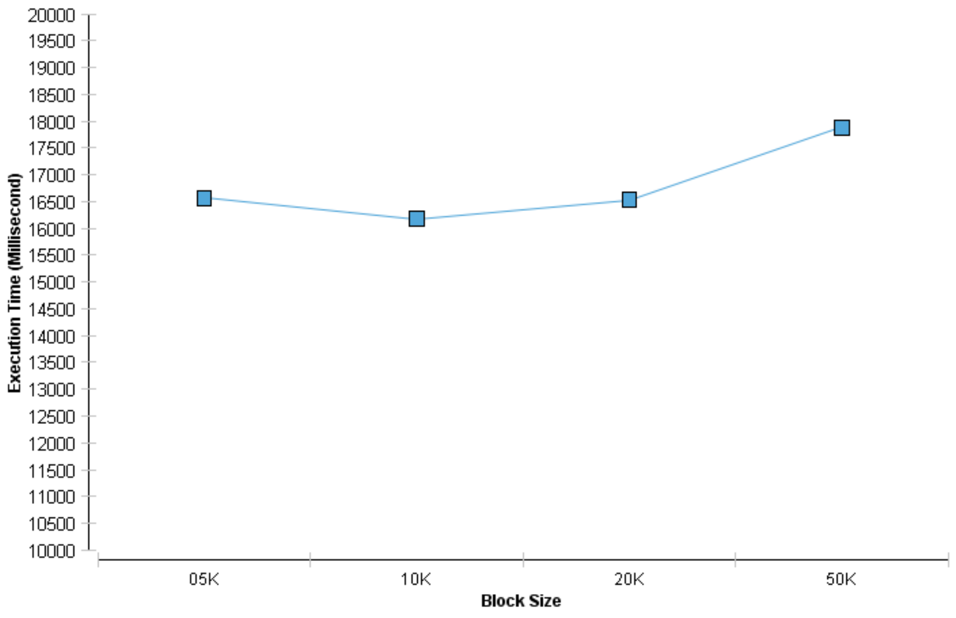
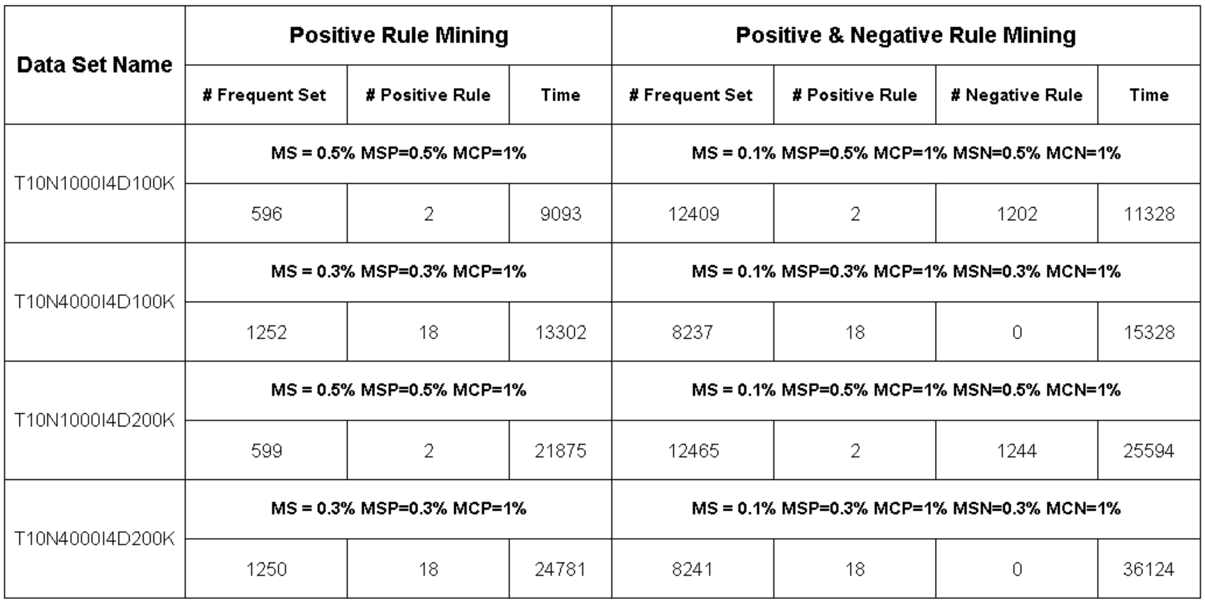


Figure 4.4: Execution Time with Different Block Size

The results show us, our proposed model gives stabile results with different block sizes. However, choosing a small size may be more advantageous, because XML processes run faster with small block size.

4.3.7. Negative Rule Mining Effect

As stated in the previous studies in literature [4, 11, 13, 14, 17], negative association rule mining is a more difficult task than positive association rule mining. The first reason of this complexity is that in negative rule mining process there is a huge frequent item sets, so finding these rules takes much more time than finding positive ones. The aim of this test to prove this declaration. We run the PNRMXS algorithm to mine only positive and both positive and negative rules.



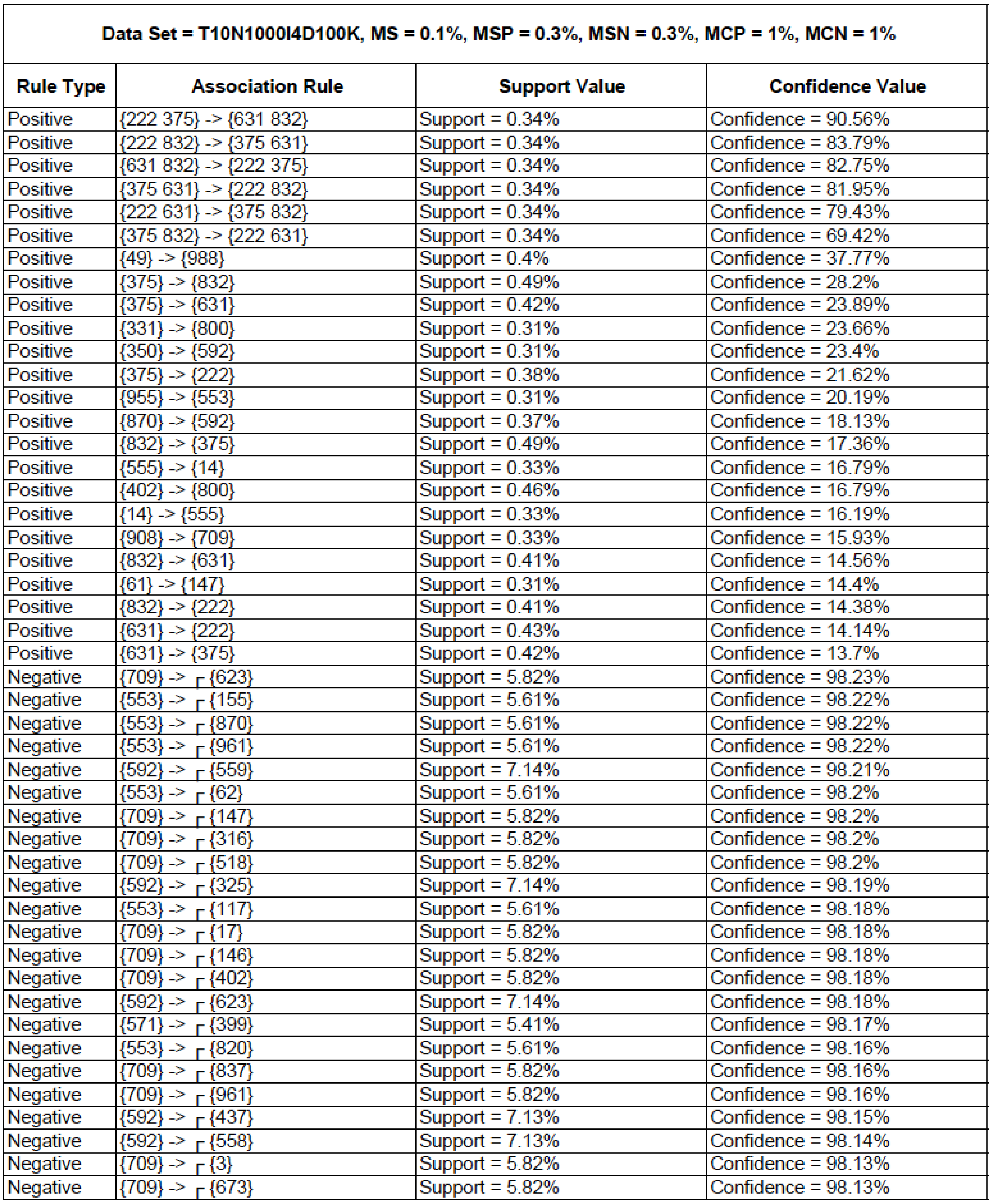
Data Set Name	**Positive Rule Mining**			**Positive & Negative Rule Mining**			
	# Frequent Set	**# Positive Rule**	**Time**	**# Frequent Set**	**# Positive Rule**	**# Negative Rule**	**Time**
T10N1000I4D100K	**MS = 0.5% MSP=0.5% MCP=1%**			**MS = 0.1% MSP=0.5% MCP=1% MSN=0.5% MCN=1%**			
	596	2	9093	12409	2	1202	11328
T10N4000I4D100K	**MS = 0.3% MSP=0.3% MCP=1%**			**MS = 0.1% MSP=0.3% MCP=1% MSN=0.3% MCN=1%**			
	1252	18	13302	8237	18	0	15328
T10N1000I4D200K	**MS = 0.5% MSP=0.5% MCP=1%**			**MS = 0.1% MSP=0.5% MCP=1% MSN=0.5% MCN=1%**			
	599	2	21875	12465	2	1244	25594
T10N4000I4D200K	**MS = 0.3% MSP=0.3% MCP=1%**			**MS = 0.1% MSP=0.3% MCP=1% MSN=0.3% MCN=1%**			
	1250	18	24781	8241	18	0	36124

Figure 4.5: Negative Rule Mining Effect on Execution Time

The results in Figure 4.5 shows the search space difference between positive and negative rule mining. In negative rule mining problem, search space is bigger 6 to 20 times than positive mining. Consequently, the execution of the algorithm takes more time in negative rule mining problem. Also, we would like to indicate one last point. As a result of our two tests, the number of negative association rules is 0. The reason of this result is that there is no negative correlated frequent item sets in test data sets.

4.3.8. Example Rule Set

In this section, we present some part of positive and negative rule sets that are mined with PNRMXS algorithm in Figure 4.6. As seen in figure, the first column shows the rule type as it is negative or positive. In the second column, the association rules with transformed items are shown. The last two columns contain the support and confidence values. This figure, also, shows the structure of our output file.



Data Set = T10N1000I4D100K, MS = 0.1%, MSP = 0.3%, MSN = 0.3%, MCP = 1%, MCN = 1%			
Rule Type	**Association Rule**	**Support Value**	**Confidence Value**
Positive	{222 375} -> {631 832}	Support = 0.34%	Confidence = 90.56%
Positive	{222 832} -> {375 631}	Support = 0.34%	Confidence = 83.79%
Positive	{631 832} -> {222 375}	Support = 0.34%	Confidence = 82.75%
Positive	{375 631} -> {222 832}	Support = 0.34%	Confidence = 81.95%
Positive	{222 631} -> {375 832}	Support = 0.34%	Confidence = 79.43%
Positive	{375 832} -> {222 631}	Support = 0.34%	Confidence = 69.42%
Positive	{49} -> {988}	Support = 0.4%	Confidence = 37.77%
Positive	{375} -> {832}	Support = 0.49%	Confidence = 28.2%
Positive	{375} -> {631}	Support = 0.42%	Confidence = 23.89%
Positive	{331} -> {800}	Support = 0.31%	Confidence = 23.66%
Positive	{350} -> {592}	Support = 0.31%	Confidence = 23.4%
Positive	{375} -> {222}	Support = 0.38%	Confidence = 21.62%
Positive	{955} -> {553}	Support = 0.31%	Confidence = 20.19%
Positive	{870} -> {592}	Support = 0.37%	Confidence = 18.13%
Positive	{832} -> {375}	Support = 0.49%	Confidence = 17.36%
Positive	{555} -> {14}	Support = 0.33%	Confidence = 16.79%
Positive	{402} -> {800}	Support = 0.46%	Confidence = 16.79%
Positive	{14} -> {555}	Support = 0.33%	Confidence = 16.19%
Positive	{908} -> {709}	Support = 0.33%	Confidence = 15.93%
Positive	{832} -> {631}	Support = 0.41%	Confidence = 14.56%
Positive	{61} -> {147}	Support = 0.31%	Confidence = 14.4%
Positive	{832} -> {222}	Support = 0.41%	Confidence = 14.38%
Positive	{631} -> {222}	Support = 0.43%	Confidence = 14.14%
Positive	{631} -> {375}	Support = 0.42%	Confidence = 13.7%
Negative	{709} -> ¬{623}	Support = 5.82%	Confidence = 98.23%
Negative	{553} -> ¬{155}	Support = 5.61%	Confidence = 98.22%
Negative	{553} -> ¬{870}	Support = 5.61%	Confidence = 98.22%
Negative	{553} -> ¬{961}	Support = 5.61%	Confidence = 98.22%
Negative	{592} -> ¬{559}	Support = 7.14%	Confidence = 98.21%
Negative	{553} -> ¬{62}	Support = 5.61%	Confidence = 98.2%
Negative	{709} -> ¬{147}	Support = 5.82%	Confidence = 98.2%
Negative	{709} -> ¬{316}	Support = 5.82%	Confidence = 98.2%
Negative	{709} -> ¬{518}	Support = 5.82%	Confidence = 98.2%
Negative	{592} -> ¬{325}	Support = 7.14%	Confidence = 98.19%
Negative	{553} -> ¬{117}	Support = 5.61%	Confidence = 98.18%
Negative	{709} -> ¬{17}	Support = 5.82%	Confidence = 98.18%
Negative	{709} -> ¬{146}	Support = 5.82%	Confidence = 98.18%
Negative	{709} -> ¬{402}	Support = 5.82%	Confidence = 98.18%
Negative	{592} -> ¬{623}	Support = 7.14%	Confidence = 98.18%
Negative	{571} -> ¬{399}	Support = 5.41%	Confidence = 98.17%
Negative	{553} -> ¬{820}	Support = 5.61%	Confidence = 98.16%
Negative	{709} -> ¬{837}	Support = 5.82%	Confidence = 98.16%
Negative	{709} -> ¬{961}	Support = 5.82%	Confidence = 98.16%
Negative	{592} -> ¬{437}	Support = 7.13%	Confidence = 98.15%
Negative	{592} -> ¬{558}	Support = 7.13%	Confidence = 98.14%
Negative	{709} -> ¬{3}	Support = 5.82%	Confidence = 98.13%
Negative	{709} -> ¬{673}	Support = 5.82%	Confidence = 98.13%

Figure 4.6: Example Rule Set

5. DISCUSSIONS & CONCLUSION

In this research, we have proposed a model for positive and negative association rule mining on XML data stream in database as a service concept that we called it PNRMXS.

Data stream mining is one of the most intensely investigated and challenging research field. Due to the characteristics of streaming data, traditional mining algorithms is not applicable, so recently researches focus on this topic. The other challenging research area in data mining is negative association rule mining. The finding negative relationships provide the data owner to review their strategy for achieving the best. However, because of its novelty and difficulty, there a few studies in negative rule mining topic. Implementing an association rule mining model in database as a service concept reveals some security problems. Organizations consider their private data as a valuable asset. Therefore, in the case of with working third party service providers, the security issues should be carefully considered.

In this study, we tried to combine the challenging areas that mentioned in previous paragraph in one model. The processes in PNRMXS model is made by two sides. In data owner side, some pre-preprocessing operations are made on data set. In service provider side the mining operations, that is the main part of our model, is run. In order to meet the necessary requirements, first we have adapted the original FP-Growth algorithm to be able to make negative rule and stream mining. In data stream mining issue, because of rapid data flow, mining operations should be made as fast as possible. However, in the negative rule mining case, there is a huge search space and this has a negative effect on algorithm performance. Therefore, we have adapted some other pruning thresholds to decrease the search space for negative rule mining.

In our experiments, various tests have been made with different parameter settings and different synthetic data sets to show the scalability and stability of our proposed model. These results have revealed that our model is fast, efficient and applicable for data stream mining domain. To the best of our knowledge, PNRMXS is an unique approach in the literature, so there was no chance for one-to-one comparison with any other rule mining model. However, in order to give an idea about the performance of PNRMXS, we have compared the some part of it with famous data stream mining model Moment [27]. There are differences between these two models. Moment algorithm finds closed frequent item sets which causes sharp decrease in item set search space and there in no rule generation process in this approach. Also, this model uses Sliding Windows data processing model. On the contrary, PNRMXS works with all frequent item set and at the same time makes positive and negative rule generation processes. Also, our data processing model is Landmark Windows model. Because of these reasons, we only compared the finding frequent set process of these two approaches. The results of the experiment have shown that the performance values of our model is applicable.

In the next section, some other topics which we think to be open for improvement will be addressed.

6. FUTURE WORKS

Although in this research, we proposed an algorithm for positive and negative association rule mining on XML data stream in database as a service concept, there is a great deal of scope for further development of this algorithm.

One possible extension of our proposed model would be an development of a more compact tree structure to decrease the memory usage and tree traversing time. Since, the efficient memory usage of data mining processes in streaming environment is a vital issue.

Another improvement can be made on the pruning strategy. As mentioned in our study, in negative rule mining there is a huge search space and this is the main difficulty of this topic. Therefore, the researches especially focus on the different pruning techniques. In order to make better pruning, we add some extra pruning thresholds in our proposed model. Although adding these thresholds, our test results show that our model still generates many negative association rules. May be with some extra improvements in the pruning techniques can give us the better results.

The PNRMXS model makes the rule mining processes on binary variables. As explained in Section 2.1, there are other types of association rules like quantitative and fuzzy variables. The other further research topic in our proposed model can be the mining of quantitative or fuzzy association rules.

One of the challenging point in our proposed model is making the rule mining process in database as a service concept. Although, this concept has many advantages, it reveals the data security problem. In order to handle this problem, we have used one-to-one mapping strategy to hide the content of the data. Also, some other encryption techniques are used to protect the streaming data. The security analysis was not in the scope of this study, but we think there would be other open points in this topic.

In our proposed model, Landmark Windows streaming data model is used for data processing. However, as explained in our study, there are other types of data processing models like Sliding Windows and Damped Model. The choosing of which type of processing model to use depends on the business context. Therefore, the adaptation of our approach to different processing models can give us an opportunity to work with different business domains and it can be thought as a future research.

REFERENCES

1. Agrawal, R., Imielinski, T., and Swami, A., "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., United States, May 25 - 28, 1993.

2. Agrawal, R. and Srikant, R., "Fast Algorithms for Mining Association Rules in Large Databases", *In Proceedings of the 20th International Conference on Very Large Data Bases*, September 12 - 15, 1994.

3. Han, J., Pei, J., and Yin, Y., "Mining frequent patterns without candidate generation", *In Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, Dallas, Texas, United States, May 15 - 18, 2000.

4. Wu, X., Zhang, C., and Zhang, S., "Efficient mining of both positive and negative association rules", *ACM Trans. Inf. Syst. 22*, pp. 381-405, 2004.

5. Tan, P. N., Steinbach, M., and Kumar, V., "Association Analysis: Basic Concepts and Algorithms", *Introduction to Data Mining*, Chapter 6, pp. 327-414, Addison-Wesley, 2005.

6. Imamura, T., Dillaway, B., and Simon, E., "XML Encryption Syntax and Processing, W3C Recommendation", "http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/", December 2002.

7. Ünay, O. and Gündem, T. I., "Parallel Processing of Encrypted XML Documents in Database as a Service Concept", *Information Technology and Control, Kaunas, Technologija*, Vol. 39, No. 4, pp. 301-309, 2010.

8. Hacigümüs, H., Mehrotra, S., and Iyer, B., "Providing Database as a Service", *18th International Conference on Data Engineering (ICDE'02)*, pp. 0029, 2002.

9. Clifton, C. and Marks, D., "Security and privacy implications of data mining", *In SIGMOD Workshop on Data Mining and Knowledge Discovery*, 1996.

10. Agrawal, R. and Srikant, R., "Privacy-preserving data mining", *In SIGMOD*, 2000.

11. Savasere, A., Omiecinski, E., and Navathe, S., "Mining for strong negative associations in a large database of customer transactions", *In Proceedings of the Fourteenth International Conference on Data Engineering*, pp. 494-502, 1998.

12. Ma, W. M. and Liu, Z. P. , "Two revised algorithms based on apriori for mining association rules", *Machine Learning and Cybernetics, 2008 International Conference on*, vol.1, pp. 350-355, July 12-15, 2008.

13. Antonie, M. L. and Zaiane, O. R., "Mining positive and negative association rules: an approach for confined rules", *In Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, 2004.

14. Yuan, X., Buckles, B. P., Yuan, Z., and Zhang, J., "Mining negative association rules," *Computers and Communications, Proceedings. ISCC 2002. Seventh International Symposium on*, pp. 623-628, 2002.

15. Hsueh, S. C., Lin, M. Y., and Lu, K. L., "Mining Generalized Association Rules for Service Recommendations for Digital Home Applications", *Intelligent Information Hiding and Multimedia Signal Processing, IIHMSP 2007, Third International Conference on*, Vol. 1, pp. 631-634, November 26-28, 2007.

16. Zhu, Y. and Shasha, D., "StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time", *Proceedings of International Conference on Very Large Database*, pp. 358-369, 2002.

17. Zhu, H. and Xu, Z., "An Effective Algorithm for Mining Positive and Negative Association Rules", *Computer Science and Software Engineering, 2008 International Conference on*, Vol. 4, pp. 455-458, December 12-14, 2008.

18. Wu, H., Lu, Z., Pan, L., Xu, R., and Jiang, W., "An Improved Apriori-based Algorithm for Association Rules Mining", *Fuzzy Systems and Knowledge Discovery. Sixth International Conference on*, Vol. 2, pp. 51-55, August 14-16, 2009.

19. Li, H. F., Lee, S. Y., and Shan, M. K., "Online mining (recently) maximal frequent itemsets over data streams", *Research Issues in Data Engineering: Stream Data Mining and Applications, RIDE-SDMA 2005. 15th International Workshop on*, pp. 11-18, April 3-4, 2005.

20. Wikipedia, "*Synthetic data*", http://en.wikipedia.org/wiki/Synthetic_data, December 2009.

21. Xiao, X. and Tao, Y., "Anatomy: Simple and effective privacy preservation", *In VLDB*, 2006.

22. Srikant, R. and Agrawal, R., "Mining quantitative association rules in large relational tables", *SIGMOD Rec. 25*, June 1996.

23. Reusch, B., "Computational Intelligence, Theory and Applications", *International Conference, 7th Fuzzy Days*, Dortmund, Germany, October 13, 2001.

24. Artamonova, I. I., Frishman, G., and Frishman, D., "Applying negative rule mining to improve genome annotation", *BMC Bioinformatics*, 8, 261, 2007.

25. Bethel, C. L., Hall, L. O., and Goldgof, D., "Mining for Implications in Medical Data", *Pattern Recognition, ICPR 2006, 18th International Conference on*, Vol. 1, pp. 1212-1215, 2006.

26. Tan, P. N. and Kumar, V., "Interestingness measures for association patterns: A perspective", *In Proc. of Workshop on Postprocessing in Machine Learning and Data Mining*, 2000.

27. Chi, Y., Wang, H., Yu, P. S., and Muntz, R. R., "Moment: maintaining closed frequent itemsets over a stream sliding window", *Data Mining, ICDM '04. Fourth*

IEEE International Conference on, pp. 59-66, November 1-4, 2004.

28. Jiang, N. and Gruenwald, L., "Research Issues in Data Stream Association Rule Mining", *ACM SIGMOD Record*, Vol. 35, No. 1, 2006.

29. Rivest, R. L., Shamir, A., and Adleman, L., "A method for obtaining digital signatures and public-key cryptosystems", *Commun. ACM 21*, pp. 120-126, February 1978.

30. Wikipedia, "*RSA*", http://en.wikipedia.org/wiki/RSA

31. The FactPoint Group, "*Leading Practices in Market Basket Analysis*", http://www.factpoint.com/pdf2/1.pdf, 2008.

32. Wan, J. W. W. and Dobbie, G., "Extracting association rules from XML documents using XQuery", *In Proceedings of the 5th ACM international workshop on Web information and data management*, New York, USA, pp. 94-97, 2003.

33. Programmers United Develop Net, http://en.pudn.com/downloads99/sourcecode/unix_linux/detail406409_en.html

34. Win, C. N. and Hla, K. H. S., "Mining frequent patterns from XML data", *Information and Telecommunication Technologies, APSITT 2005 Proceedings 6th Asia-Pacific Symposium on*, pp. 208-212, November 10, 2005.

35. Bouloutas, A. T., Calo, S., and Finkel, A., "Alarm correlation and fault identification in communication networks", *Communications, IEEE Transactions on*, Vol. 42, pp. 523-533, 1994.

36. Wikipedia, "*Key size*", http://en.wikipedia.org/wiki/Key_size

37. Wong, W. K., Cheung, D. W., Hung, E., Kao, B., and Mamoulis, N., "Security in outsourcing of association rule mining", *In Proceedings of the 33rd International Conference on Very Large Data Bases*, pp. 111-122, 2007.

38. Yu, J. X., Chong, Z., Lu, H., and Zhou, A., "False Positive or False Negative: Mining Frequent Itemsets from High Speed Transactional Data Streams", *In Proceedings of the Thirtieth International Conference on Very Large Data Bases*, 2004.

39. Nahar, J. and Tickle, K., "Significant cancer risk factor extraction: an association rule discovery approach", *IEEE International Workshop on Data Mining and Artificial Intelligence in conjunction with 11th IEEE International Conference on Computer and Information Technology*, Khulna, Bangladesh, pp. 108-114, December 25-27, 2008.

40. Li, T. Y. and Li, X. M., "A LFP-tree based method for association rules mining in telecommunication alarm correlation analysis", *The Journal of China Universities of Posts and Telecommunications*, Vol. 14, pp. 6-9, October 2007.

41. Wikipedia, "*Advanced Encryption Standard*", http://en.wikipedia.org/wiki/Advanced_Encryption_Standard

42. W3Schools, "*XQuery*", http://www.w3schools.com/xquery/default.asp

43. Coenen, F., "The LUCS-KDD FP-growth Association Rule Mining Algorithm", *Department of Computer Science, The University of Liverpool, UK*, http://www.cxc.liv.ac.uk/∼frans/KDD/Software/FPgrowth/fpGrowth.html, 2003.

44. UCI Machine Learning Repository, "*Mushroom Data Set*", http://archive.ics.uci.edu/ml/datasets/Mushroom

45. Xu, Z. M. and Zhang, R., "Financial revenue analysis based on association rules mining", *Computational Intelligence and Industrial Applications, PACIIA 2009, Asia-Pacific Conference on* , Vol. 1, pp. 220-223, November 28-29, 2009.

46. Wan, J. W. W. and Dobbie, G., "Mining association rules from XML data using XQuery", *In Proceedings of the second workshop on Australasian Information Security, Data Mining and Web Intelligence, and Software Internationalisation*, Vol.

32. Australian Computer Society, Inc., Darlinghurst, Australia, pp. 169-174, 2004.

47. Wu, M. C., Lin, S. Y., and Lin, C. H., "An effective application of decision tree to stock trading", *Expert Systems with Applications*, Vol. 31, Issue 2, pp. 270-274, August 2006.