Secure Frequent Itemset Hiding Techniques in Data Mining

Arpit Agrawal[1]
Asst. Professor Department of Computer Engineering
Institute of Engineering & Technology
Devi Ahilya University
M.P., India

Jitendra Soni
Asst. Professor Department of Computer Engineering
Institute of Engineering & Technology
Devi Ahilya University
M.P., India

Abstract
In privacy preserving data mining (PPDM) field of research studies how knowledge or patterns can be extracted from large data stores while maintaining commercial or legislative privacy constraints. Association rules (frequent itemsets), classification and clustering are main methods used in data mining research. One of the great challenges of data mining is finding hidden patterns without violating data owners' privacy. Privacy preserving data mining came into prominence as a solution. In the aim of the paper, Matrix Apriori algorithm is modified and a frequent itemset hiding framework is developed. Four frequent itemset hiding algorithms are proposed such that: first all versions work without pre-mining so privacy breech caused by the knowledge obtained by finding frequent itemsets is prevented in advance, secondly efficiency is increased since no pre-mining is required, thirdly supports are found during hiding process and at the end sanitized dataset and frequent itemsets of this dataset are given as outputs so no post-mining is required, finally the heuristics use pattern lengths rather than transaction lengths eliminating the possibility of distorting more valuable data.

Keywords: *Data Mining, Frequent Item Mining, Data Hiding, PPDM.*

1. Introduction

Privacy preserving data mining is divided into two major categories: data hiding and rule hiding. Data hiding aims to design new protocols to perturb, anonymize or encrypt raw data while sensitive private data is protected and underlying patterns can still be discovered. Rule hiding refers to design algorithms is such a way that sensitive rules or patterns stay unrevealed while remaining rules or patterns can still be mined. The original data is distorted or blocked by rule hiding algorithms. Privacy, the new direction of data mining research is the main motivation for start point of this paper study. It is decided to apply privacy preserving data mining techniques for frequent itemset mining.

Surveying literature, it has been seen that many algorithms for association rule or frequent itemset hiding are Apriori based and as it is mentioned above it has a disadvantage of multiple database scanning. Therefore, algorithms without candidate generation are studied firstly. Data mining is a growing area of study in computer science and it is applied to many fields. However, malicious usage may cause privacy problems. It is a challenge to perform data mining without violating privacy of data or knowledge [1-3]. This necessity emerged privacy preserving data mining. It is a recently grown aspect of data mining and there is much work to do. Attracted by these and popularity of frequent itemset mining in data mining, frequent itemset hiding of privacy preserving data mining is studied in this paper. The aims of this paper are: To understand frequent itemset mining and compare two of algorithms Matrix Apriori and FP-Growth working without candidate generation. To understand privacy preserving data mining and frequent itemset hiding and propose frequent itemset hiding algorithm.

2. Background

Data Mining
Data mining is a recently emerging field, connecting the three worlds of databases, artificial intelligence and statistics. It involves the use of data analysis tools to discover previously unknown, valid patterns and relationships in large datasets. Model created for data mining can be predictive or descriptive. Predictive models make a prediction about values of data using known results found from different data. Descriptive models identify patterns of relationships in data. Common tasks of predictive models are classification, regression, time series analysis and prediction. Clustering, summarization, association rules and sequence discovery are common tasks of predictive data mining models. These are depicted in Figure 1.

ISSN: 2278 – 1323

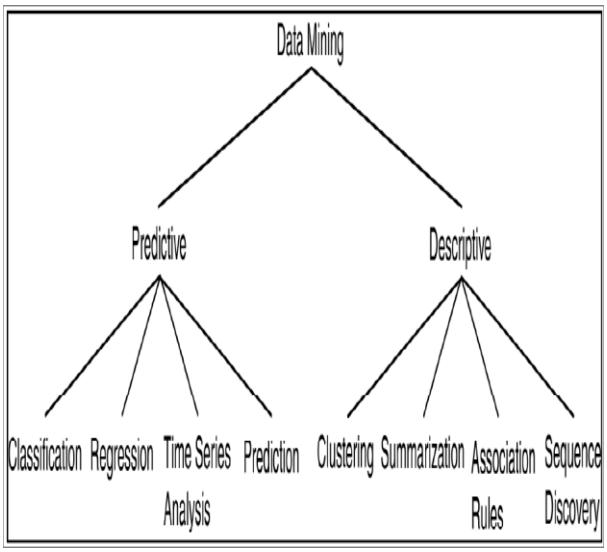

Figure 1 Data mining models and tasks

There are mainly three data mining techniques: classification, clustering and association rule mining. Classification uses a training set and builds a classifier to predict the classes of new instances. Clustering divides dataset into clusters of which members are similar to each other and different from members of other clusters. Association rule mining finds patterns and relationships among dataset. These techniques are briefly introduced in following subsections [2-4].

Classification

Classification maps data into predefined groups or classes. Simply classifies data based on training set and uses it classifying new data. Classification algorithms can be divided into five as statistical-based, distance-based, decision tree based, neural network-based and rule based algorithms. It is formally defined as: Given a database D = {t1, …, tn} of tuples (items, records) and a set of classes C={C1,… Cm}, the classification problem is to define a mapping f: D→C where each ti is assigned to one class. A class, Cj, contains precisely those tuples mapped to it; that is, Cj= {ti|f(ti)=Cj, 1 <= i <= n, and ti ∈ D} [3-6].

Clustering

The process of grouping a set of physical or abstract objects into classes of similar objects is called clustering. A cluster is a collection of data objects that are similar to one another within the same cluster and are dissimilar to the objects in other clusters. Unlike classification the groups are not pre defined. Clustering algorithms can be divided into three as hierarchical, partitioned, categorical algorithms. Formal definition of clustering is:

Given a database D= {t1, t2,…,tn} of tuples and an integer value k, the clustering problem is to define a mapping f: D→ {1,…,k} where each ti is assigned to one cluster Kj, 1 <= j<= k. A cluster, Kj, contains precisely those tuples mapped to it; that is, Kj= {ti|f(ti) = Kj, 1 <= I <= n, and ti ∈ D} [1] and [4-7].

Association Rule Mining

Association rule miming finds relationships and patterns between items in a database. It is a two step process. Firstly, frequent itemsets are found and secondly from these itemsets, rules are produced. Formal definition of association rule mining is Given: a set of items I={I1, I2, …, Im} and a database of transactions D={t1,t2,…,tn} where ti={Ii1, Ii2, …,Iik} and Iij ∈ I and X,Y are set of items, the association rule problem is to identify all association rules X → Y with a minimum support and confidence where support of association rule X →Y is the percentage of transactions in the database that contain X U Y and confidence is the ratio of support of X U Y to support of X [8-9].

Frequent Itemset Mining

The progress in bar-code and computer technology has made it possible to collect data about sales and store as transactions which is called basket data. This stored data attracted researches to apply data mining to basket data. As a result association rules mining came into prominence which is mentioned as synonymous to market basket analysis. As stated before association rule mining is a two step process. Firstly, frequent itemsets are found using minimum support value, and this step is the main concentration of association rule mining algorithms. Later from these itemsets using minimum confidence value rules are produced. As the differing part of the algorithms are frequent itemset finding part, association rule mining, frequent itemset mining or frequent pattern mining terms are used interchangeably. Association rule mining which was first mentioned is one of the most popular data mining approaches. Not only in market business but also in variety of areas association rule mining is used efficiently. In Apriori algorithm is used on a diabetic database and developed application is used to discover social status of diabetics and represents a survey of frequent pattern mining from gene expression data. In a report, association rules are listed in the success stories part and in a survey the Apriori algorithm is listed in top 10 data mining algorithms.

There have been many improvements for Apriori algorithm. Sampling approach is proposed vertical data format for clustering transactions and producing frequent itemsets from these clusters. Although these algorithms are showed to perform better than Apriori,

ISSN: 2278 – 1323

most significant improvement is lately proposed FPGrowth algorithm. The main objective is to skip candidate generation and test step which is the bottleneck of the Apriori like methods. The algorithm uses a compact data structure called FP-tree and pattern fragment growth mining method is developed based on this tree. FP-growth algorithm scans database only twice. It uses a divide and conquers strategy.
In several extensions for both Apriori and FP-growth accuracy of results is sacrificed for better speed. Matrix Apriori proposed in, combines positive properties of these two algorithms. Algorithm employs two simple structures: A matrix of frequent items called MFI and a vector storing the support of candidates called STE. Matrix Apriori consists of three procedures. First builds matrix MFI and populates vector STE. Second modifies matrix MFI to speed up frequent pattern search. Third identifies frequent patterns using matrix MFI and vector STE [2] and [5-7].

Data Hiding
The main objective of data hiding is to design new protocols to perturb, anonymize or encrypt raw data so that sensitive data remains sensitive during and after the mining operation while underlying data patterns can still be discovered. In the following subsections, techniques used in data hiding are introduced.

Perturbation
One approach to privacy-preserving data mining is based on perturbing the original data, then providing the perturbed dataset as input to the data mining algorithm. The privacy-preserving properties are a result of the perturbation. Data values for individual entities are distorted, and thus individually identifiable (private) values are not revealed.
The randomization technique uses data distortion methods in order to create private representations of the records. In most cases, the individual records cannot be recovered, but only aggregate distributions can be recovered. These aggregate distributions can be used for data mining purposes. Two kinds of perturbation are possible with the randomization method: Additive Perturbation: In this case, randomized noise is added to the data records. The overall data distributions can be recovered from the randomized records. Data mining and management algorithms redesigned to work with these data distributions. Multiplicative Perturbation: In this case, the random projection or random rotation techniques are used in order to perturb the records.
Rule Hiding
The main focus of rule hiding is association rules and frequent patterns. Association rule hiding refers to the process of modifying the original database in such a way that certain sensitive association rules disappear without seriously affecting the data and the non-sensitive rules. The main goal here is to hide as many sensitive rules as possible, while keeping preserved as many non-sensitive rules as possible.
To make the necessity of hiding association rules clear here is a scenario. Let us suppose that we are negotiating with Dedtrees Paper Company, as purchasing directors of Big-Mart, a large supermarket chain. They offer their products in reduced prices, provided that we agree to give them access to our database of customer purchases. We accept the deal and Ded-trees starts mining our data. By using an association rule mining tool, it can be found that people who purchase skim milk also purchase Green Paper. Ded-trees now runs a coupon marketing campaign offering a 50 cents discount on skim milk with every purchase of a Ded-trees product. The campaign cuts heavily into the sales of Green Paper, which increases the prices to us, based on the lower sales. During our next negotiation with Ded-trees, we found out that with reduced competition they are unwilling to offer to us a low price. Finally, we start losing business to our competitors, who were able to negotiate a better deal with Green Paper. In other words, the aforementioned scenario indicates that Big-Mart should sanitize competitive information (and other important corporate secrets of course) before delivering their database to Ded-trees, so that Ded-trees does not monopolize the paper market.
As seen in the example above hiding association rules which are sensitive is a should be considered subject in information sharing for data mining. Distortion and blocking are techniques used in rule hiding and introduced in following subsections [8-10].

3. Proposed Techniques

PPDM has two aspects as input and output privacy. To protect input privacy, data hiding techniques are applied such that data mining can still be done without violating private individual data. To protect output privacy, rule or knowledge hiding techniques are applied. These techniques ensure that private rules or patterns which can be extracted from given data are hidden while remaining ones can still be mined. Rule hiding is concentrated on association rules and frequent itemsets. Algorithms operate to distort items in transactions of database in such a way that as many as sensitive itemsets or association rules

ISSN: 2278 – 1323

hidden and as many as non-sensitive itemsets or association rules extracted.

Many approaches for frequent itemset hiding are Apriori based and needs multiple database scans. Besides, these techniques require pre-mining to calculate support of sensitive itemsets. Therefore, it is decided to propose such algorithm that it avoids multiple database scans and pre-mining of frequent patterns. Matrix Apriori, a two database scan frequent itemset mining algorithm, is used for proposed itemset hiding algorithm. The approach does not require pre-mining and supports are calculated during hiding process. In addition, four distortion strategies are proposed which use pattern lengths instead of transaction lengths for item selection. Now, firstly, frequent itemset mining is introduced and two algorithms FP-Growth and Matrix Apriori are explained and demonstrated to be self contained for itemset hiding section. Following, proposed Matrix Apriori based frequent itemset hiding algorithms are explained.

Frequent Itemset Mining

Association rule mining was first introduced by Agrawal the popular Apriori algorithm was proposed. It computes the frequent itemsets in the database through several iterations. Each iteration has two steps: candidate generation and candidate selection. Database is scanned at each iteration. Apriori algorithm uses large itemset property: any subset of a large itemset must be large. Candidate itemsets are generated as supersets of only large itemsets found at previous iteration. This reduces the candidate itemset number. Among many versions of Apriori, FP-Growth has been proposed in association rule mining research with the idea of finding frequent itemsets without candidate generation (Han 2000). FP-Growth uses tree data structure and scans database only twice showing notable impact on the efficiency of itemset generation phase. Lately an approach named Matrix Apriori is introduced with the claim of combining positive properties of Apriori and FP-Growth algorithms. In this approach, database is scanned twice as in the case of FP-Growth and matrix structure used is simpler to maintain.

FP-Growth Algorithm

The FP-Growth method adopts a divide and conquers strategy as follows: compress the database representing frequent items into a frequent-pattern tree, but retain the itemset association information, and then divide such a compressed database into a set of condition databases, each associated with one frequent item, and mine each such database.

The algorithm is given in Figure 2. Firstly database is read and frequent items are found which are the items are occurring in transactions less than minimum support. Secondly database is read again to build FP-tree. After creating the root, every transaction is read in an ordered way and pattern of frequent items in the transaction is added to FP-tree and nodes are connected to frequent items list and each other. This interconnection makes frequent pattern search faster avoiding the traversing of the entire tree. When considering the branch to be added for a transaction, the count of each node along a common prefix is incremented by 1. Nodes of same items are interconnected where most left one is connected to item in frequent items list. If the prefix of branch to be added does not exists then it is added as a new branch to root. After constructing the tree the mining proceeds as follows. Start from each frequent length-1 pattern (frequent item), construct its conditional pattern base, then construct its conditional FP-tree and perform mining recursively on such a tree. The support of a candidate (conditional) itemset is counted traversing the tree. The sum of count values at least frequent item's nodes (base node) gives the support value.

Input: Database D, Minimum Support minsup
Output: Frequent itemset FIs
Begin
1 Read D
2 Count item occurrences
3 Exclude item below minsup
4 Create root FP tree
5 Read D
6 For every transaction read frequent item in descending order as in FI-list
7 if first item has already connected to root increase node's count and set as current node
8 Else add new mode to root and set as current node and connect to FI-list
9 For remaining items
10 If item exists in one of child nodes set it as current node and increase count
11 else add item as new node and set it as current node
12 if there is node connected to FIlist connect to last node of that item
13 Else connect to FIlist
14 End
15 End
16From least frequent item to most frequent item in FIlist
17 From I=2 to FIlist length
18 Generate candidate length (I) itemset by extending item with other items in FIlist
19 Count support by traversing base nodes of item
20 If support >= minsup give itemset as output
21 Continue extending itemset and go to line 18
22 Else stop extending and go to line 17
23 End
24 End
25 End

Figure 2 FP-Growth Algorithms

ISSN: 2278 – 1323

Matrix Apriori Algorithm

Matrix Apriori is similar to FP-Growth in the database scan step. However, the data structure build for Matrix Apriori is a matrix representing frequent items (MFI) and a vector holding support of candidates (STE). The search for frequent patterns is executed on this two structures, which are easier to build and use compared to FP-tree.

Matrix Apriori algorithm is given. Firstly database is read and frequent items are found which are the items are occurring in transactions less than minimum support. Secondly database is read again to build MFI and STE. Following this, a second scan on database is executed. During the scan the MFI and STE is built as follows. Each transaction is read. If the transaction has any item that is in the frequent item list then it is represented as "1" and otherwise "0". This pattern is added as a row to MFI matrix and its occurrence is set to 1 in STE vector. While reading remaining transactions if the transaction is already included in MFI then in STE its occurrence is incremented. Otherwise it is added to MFI and its occurrence in STE is set to 1. After reading transactions, the MFI matrix is modified to speed up frequent pattern search. For each column of MFI, beginning from the first row, the value of a cell is set to the row number in which the item is "1". If there is not any "1" in remaining rows then the value of the cell is set to "1" which means down to the bottom of the matrix, no row contains this item. After constructing the MFI matrix, finding patterns is simple. Beginning from the least frequent item, create candidate itemsets and count its support value. The support value of an itemset is the sum of the items at STE of which index are rows where all the items of the candidate itemset are included in MFI's related row.

Discussion on FP-Growth and Matrix Apriori Algorithms

It will be beneficial to give a short comparison of given algorithms with an example to show the execution of the algorithms. First scans of both algorithms are carried out in the same way. Frequent items are found and listed in order. During second scan, FP-Growth adds transactions to tree structure and Matrix Apriori to matrix structure. Addition of a transaction to the tree structure needs less control compared to matrix structure. For example, consider 2nd and 3rd transactions. Second transaction is added as a branch to the tree and as a row to the matrix. But addition of third transaction shows the difference. For tree structure we need to control only the branch that has the same prefix with our transaction. So addition of a new branch to node E is enough. On the other hand, for the matrix structure we need to control all the items of rows. If we find the same pattern then we increase the related item of STE. Otherwise we need to scan matrix till we find the same pattern. If we cannot find then a new row is added to matrix. It seems that building matrix needs more control and time, however, management of matrix structure is easier compared to tree structure. Finding patterns for both algorithms need producing candidate itemsets and control. This is called conditional pattern base in FP-Growth and there is no specific name for Matrix Apriori. Counting support value is easy to handle in Matrix Apriori by sequentially top down sum of related rows of STE. However, in FP-Growth counting support is complex by traversing the tree, selecting related nodes and sum values in selected nodes.

Frequent Itemset Hiding

One of the prominence techniques in data mining is frequent itemset or association rule mining however; obtained outputs may cause violation of knowledge privacy. There may be some situations where knowledge extracted by rule mining algorithms includes rules or itemsets that should stay unrevealed. These itemsets are called sensitive itemsets. Itemset hiding intends to modify database in such a way that sensitive itemsets are hidden with minimum side effects on non-sensitive ones. The first study on rule hiding shows that sanitization of the database is NP-Hard and heuristic approaches are needed. Heuristic approaches are based on support and confidence reduction. Following studies propose algorithms for itemset hiding and association rule hiding respectively. These algorithms distort items in the database. However, there may be such conditions that writing false values may cause problems. The approach used in unknown values instead of writing false values on the database. Many itemset or rule hiding approaches are based on Apriori algorithm which needs multiple database scans and pre-mining of association rules. On the other hand FP-Growth algorithm, which has a better performance compared to Apriori, makes two database scans for finding frequent itemsets. The work presented to uses hiding algorithm based on P-tree similar to FP-tree of FPGrowth algorithm. They sanitize informative rules and eliminate need for pre-mining of association rules. Another, frequent itemset mining algorithm with two database scans is Matrix-Apirori. It is simpler than FP-Growth in terms of maintenance of the compact data structure and performs better which leads to propose itemset hiding algorithms. Proposed algorithms for frequent itemset hiding are explained and demonstrated in following.

ISSN: 2278 – 1323

Matrix Apriori Based Frequent Itemset Hiding Algorithms

As displayed in Figure 3, proposed privacy preserving frequent itemset mining approach gets dataset *D,* sensitive itemsets *Ls* and minimum support *minsup* as input and returns sanitized dataset *Ds* with frequent itemsets which can be found from *Ds* as *FIs*. Sensitive itemsets are given without any knowledge if those itemsets are frequent or not. If any itemset given as sensitive is frequent in original database then it is hidden through itemset hiding process. Most hiding approaches first do mining and calculate support of all frequent itemsets then start hiding process. This has two disadvantages i) it might cause a privacy breech if the one performing hiding process is not trusted because all frequent itemsets are required to be known before the hiding process and ii) it requires pre-mining causing decrease in efficiency. Proposed approach ensures that user does not know whether given sensitive itemset was frequent in original dataset because frequent itemsets are found during hiding process and eliminates the need for pre-mining process.

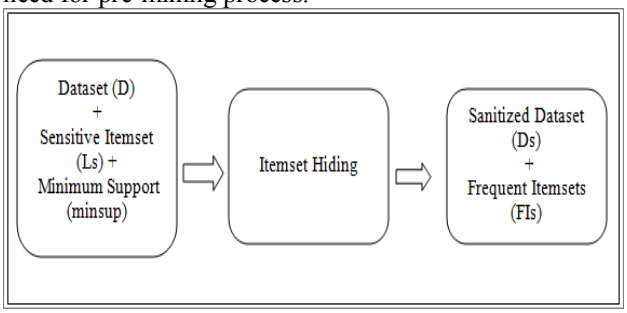

Figure 3 Sanitization framework

Itemset hiding process is based on the Matrix Apriori algorithm. Matrix Apriori is a frequent itemset mining algorithm without candidate generation and scans database only twice. At first scan for the specified minimum support frequent items are found. At second scan, matrix data structure called MFI and support holder vector STE is build. For every transaction in database, the pattern consists of frequent items is read and added to MFI matrix and occurrences are updated on STE vector. Frequent itemset mining is done on this compact data structure which eliminates the need for database scan for itemset support counting Matrix Apriori algorithm is modified in proposed approach to have capabilities for itemset hiding. As stated in line 1, database *D* is read and matrix data structure *MFI* and *STE* is build. This time, while building *MFI* and *STE*, we also construct a transaction list as *TList* which keeps the transaction ids of transactions containing the itemset in each row of *MFI*. In proposed approach, transaction selection for modifying is done on *MFI* and database scan in order to find transaction is eliminated. Between lines 2 and 14 for every itemset in sensitive itemsets list *Ls*, hiding process is run. Support value for sensitive itemset is calculated using *MFI* and *STE*. If the support of the itemset is above *minsup* then the number of iterations to hide itemset is calculated (line 4). This number indicates number of distortions to be done on the dataset to reduce the support of the sensitive itemset *Is* below *minsup*. Following this, at each iteration transaction to modify is selected (lines 6 and 7). There are two strategies for transaction selection. First one is to find shortest pattern for transaction selection. First one is to find shortest pattern that include the sensitive itemset and select the last transaction from *TList*. Second strategy is to find longest pattern and select the transaction from *TList*. Most approaches use transaction length to decide transaction to modify. However, compact matrix structure of proposed approach has more valuable information as patterns of frequent items so length of the pattern is used instead of length of the transaction. This approach also eliminates the need for database access in choosing decision.

When transaction is selected we need to select an item of the transaction for distortion (line 8). There are two strategies for selection of item to distort: *maxFI* and *minFI*. Using *maxFI*, most frequent item of sensitive itemset is distorted on transaction. If *minFI* is used then least frequent item of sensitive itemset is distorted on transaction. Selected item is distorted in transaction (line 9), the distortion technique is replacing "1" with "0" in related cell. Matrix structure *MFI* is updated after distortion (line 10). We decrease the value of related row in *STE* (line 11) and delete transaction modified in that row of *TList* (line 12). By this way it is ensured that we have compact mirror of semisanitized dataset in *MFI*, *STE* and *TList* throughout the hiding process.

The selection and distortion process is repeated until the support of sensitive itemset *Is* is below *minsupport*. After sanitization of a *Is* the next itemset is read from *Ls* and sanitized. At final step (line 16) frequent itemsets *FIs* of sanitized dataset *Ds* are found using up-to-date *MFI* and *STE*.

Input: Original Database D, minimum support minsup, List of sensitive itemset Ls
Output: Sanitized Database Ds, Frequent itemset of Ds FIs
Begin
1 Read D and build MFI, STE and Tlist
2 For every itemset in Ls
3 Calculate support of the sensitive itemset Is
*4 Number of iterations = (Support of Is – minsup) * number of transactions in Tlist +1*
5 For 1 to number of iterations
6 select pattern from MFI (Support or longest one)
7 select transaction from TList

ISSN: 2278 – 1323

8 Select item to distort (most frequent MaxFI or least frequent MinFI in Is)
9 Distort item in D
10 Update MFI
11 Update STE
12 Update TList
13 End
14 End
15 Find frequent itemset using MFI FIs
16 Return Ds, FIs
17 End

Figure 4 Itemset hiding algorithm

4. Results Analysis

The large itemset property saying "any subset of large itemset must be large" made Apriori so popular that it boosted data mining research. However, it has a bottleneck that for generating candidate itemsets Apriori scans database several times. FP-Growth came up with the idea of eliminating database scan for candidate itemset generation and testing. It uses a compact data structure called FP-tree which can be thought as a summary of original database. FPGrowth scans database only twice and Matrix Apriori is another algorithm that scans database only twice. Instead of tree Matrix Apriori deploys a matrix structure which speeds up the search for frequent itemsets. Although it is claimed to be faster than FP-Growth there is no work showing performances. This is done as the first part of the study.

Knowledge extracted by frequent itemset mining may cause privacy problems if some itemsets are sensitive which means they must be remain unrevealed. Privacy preserving data mining techniques for itemset mining are developed to come over this problem. They simply distort items in transactions of database and prevent sensitive itemsets to be exracted. Many hiding techniques are Apriori based.

Now we represent the results and discussions of performance evaluations. All applications are developed using Lazarus IDE. Firstly, two frequent itemset mining algorithms working without candidate generation FP-Growth and Matrix Apriori are compared. This study formed the basis for proposing frequent itemset hiding algorithms using Matrix Apriori. Secondly, four algorithms proposed for itemset hiding are compared. This study gives not only comparison of proposed algorithms but also effects of hiding process for different cases.

Comparison of Two Frequent Itemset Mining Algorithms

Matrix Apriori and FP-Growth algorithms which were discussed in previous chapter are compared. AR-Tool dataset generator is used for our synthetic datasets. Two case studies analyzing the algorithms are carried out step by step using two synthetic datasets generated in order i) to see their performance on datasets having different characteristics, ii) to understand the causes of performance differences in different phases. In order to keep the system state similar for all test runs, we assured all back-ground jobs which consume system resources were inactive. It is also ensured that test runs give close results when repeated.

Simulation Environment for Frequent Itemset Mining Evaluations

The test runs are performed on a computer with 2.4 GHz dual core processor and 3 GB memory. At each run, both programs give results about data mining process. These are

→time cost for first scan of database,
→number of frequent items found at first scan of database,
→time cost for second scan of database and building the data structure,
→time cost for finding frequent itemsets,
→number of frequent itemsets found after mining process,
→total time cost for whole data mining process.

Although real life data has different characteristics from synthetically generated data as mentioned, synthetic data is used since the controls of parameters were easily manageable.

In the following the performance analysis on the algorithms for two case studies is given. For the generated data sets, it is aimed to observe how change of minimum support affects the performance of algorithms. The algorithms are compared for six minimum support values in the range of 20% and 28%.

Case 1: Database of Long Patterns with Low Diversity of Items

A database is generated for having long patterns and low diversity of items where number of items=10000, number of transactions=30000, average size of transactions=20, average size of patterns=10. Decrease in minimum support increases the number of frequent items from 20 to 240.

Number of frequent itemsets is in dataset while minimum support value is varied. It is clear that decrease in minimum support increases the number of frequent itemsets from 1014 to 198048.

The total performance of Matrix Apriori and FP-Growth is demonstrated as: it is seen that their performance is identical for minimum support values above 7,5%. On the other hand below 7.5% minimum

ISSN: 2278 – 1323

support value Matrix Apriori performs clearly better such that at 2.5% threshold it is 230% faster.
The reason of FP-Growth's falling behind at total performance can be understood by looking at the performance of phases of evaluation. First phase of Matrix Apriori shows similar pattern with the number of frequent items.

Case 2: Database of Short Patterns with High Diversity of Items
A database is generated for short patterns and high diversity of items using the parameters where number of items=30000, number of transaction=30000, average size of transactions=20, average size of patterns=5. Frequent items found changes from 58 to 127 with decreasing minimum support values.
The total performance of both algorithms is increase in minimum support decreases runtime for both algorithms. For minimum support values 12.5% and 15% FP-Growth performed faster by up to 56%. However, for lower minimum support values Matrix Apriori performed better.

FP-Growth and Matrix Apriori Comparison
The performance of FP-Growth and Matrix Apriori algorithms are analyzed phase by phase, when minimum support threshold is changed. Two databases with different characteristics are used for our case studies. In both case studies, performances of algorithms are observed between minimum support values of 2.5% and 15%. First case study is carried out on a database of long patterns with low diversity of items. It is seen that at 10%-15% minimum support values, performances of both algorithms are close. However, below 10% value, the performance gap between the algorithms becomes larger in favor of Matrix Apriori. Another point is that first phase of Matrix Apriori is affected from minimum support change more than FP-Growth. This is a result of increase in frequent items count. This increment affects building data structure step of Matrix Apriori dramatically. On the other hand, matrix data structure is faster leading to better total performance of Matrix Apriori.
Our second case study is performed on a database of short patterns with high diversity of items. It is seen that at 12.5%-15% minimum support values, performances of both algorithms are close. However, below 12.5% value, the performance gap between the algorithms becomes larger in favor of Matrix Apriori. It is seen that the impacts of having more items and less average pattern length caused both algorithms to have more runtime values compared to first case study. Common points in both case studies are i) Matrix Apriori is faster at finding itemset phase compared to FP-Growth and slower at building data structure phase, ii) for threshold values below 10% Matrix Apriori is more efficient by up to 30%, iii) first phase performance of Matrix Apriori is correlated with number of frequent items, iv) second phase performance of FP-Growth is correlated with number of frequent itemsets.

Comparison of Matrix Apriori Based Frequent Itemset Hiding Algorithms
In this section, performance evaluation of four versions of our itemset hiding algorithms is given. These are spmaxFI (select ***s***hortest ***p***attern and ***max***imum of ***f***requent ***i***tems in the itemset), spminFI (select ***s***hortest ***p***attern and ***min***imum of ***f***requent ***i***tems in the itemset), lpmaxFI (select ***l***ongest ***p***attern and ***max***imum of ***f***requent ***i***tems in the itemset) and lpminFI (select ***l***ongest ***p***attern and ***min***imum of ***f***requent ***i***tems in the itemset). Two synthetic databases are used to see effect of different database size. The algorithms are executed on databases i) to see effect of increasing number of sensitive itemsets, ii) to see effect of increasing support of sensitive itemset. The effects observed are number of lost itemsets as side effect, time cost for hiding process and number of items distorted for hiding itemsets. During evaluations, it is ensured that the system state is similar for all test runs and results are checked for consistency.

5. Conclusion and Future Works

Data mining aims to discover knowledge or patterns from the data especially large databases. However, there may be such situations that private data may be under violation because of gained knowledge or extracted knowledge by itself contains some private knowledge. Privacy preserving data mining arise from the need for do data mining without violating privacy of data or knowledge. Data hiding and rule hiding are two branches of PPDM. Data hiding techniques preserve the private data while rule hiding techniques preserve the private rules or patterns. The aim of this paper is to propose algorithms for privacy preserving frequent pattern mining. In conclusion, this paper shows that Matrix Apriori is a better performer compared to FP-Growth algorithm and it is an efficient way to use it for frequent itemset hiding. The main contributions of the study are i) sanitization framework eliminating the need for pre-mining and the database scan for post-mining after sanitization, ii) four versions of Matrix Apriori based frequent itemset hiding algorithms, iii) the idea of using pattern lengths for distortion strategy. As a future study, the efficiency of Matrix Apriori algorithm can be increased and may be parallelized. In addition, for

ISSN: 2278 – 1323

Matrix Apriori based frequent itemset mining algorithms we plan to carry out further evaluations on different databases, especially those having bigger average transaction lengths, to see the impact of having multiple sensitive itemsets in a single transaction on distortion.

References

[1] Gkoulalas-Divanis, A., Verykios, V. 2006. An Integer Programming Approach for Frequent Itemset Hiding. In Proceedings of the 15th ACM International Conference on Information and Knowledge Management: 748-757.

[2] Gkolalas-Divanis, A., Verykios, V. 2008. Exact Knowledge Hiding Through Database Extension. IEEE Transactions on Knowledge and Data Engineering (21): 699-713.

[3] Grossman, R., Kasif, S., Moore, R., Rocke, D., Ullman, J. 1998. Data Mining Research: Opportunities and Challenges. A Report of three NSF Workshops on Mining Large, Massive, and Distributed Data, http://pubs.rgrossman.com/dl/misc-001.pdf (last access May 31, 2010).

[4] Han, J. and Kamber, M. 2005. Data Mining: Concepts and Techniques. Morgan Kaufman.

[5] Han, J., Pei, J., Yin, Y. 2000. Mining Frequent Patterns without Candidate Generation. In Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data: 1-12.

[6] Hipp, J., Güntzer, U., Nakhaeizadeh, G. 2000. Algorithms for Association Rule Mining a General Survey and Comparison. SIGKDD Explorations Newsletter (2): 58-64.

[7] Huang, H., Wu, X., Relue, R. 2002. Association Analysis with One Scan of Databases. In Proceedings of the 2002 IEEE International Conference on Data Mining: 629-632.

[8] Kantarcioglu, M. and Clifton, C. 2004. Privacy-Preserving Distributed Mining of Association Rules on Horizontally Partitioned Data. IEEE Transactions on Knowledge and Data Engineering (16): 1026-1037.

[9] Oliveira, S.R.M., Zaiane, O.R.: Privacy preserving clustering by data transformation. In: 18th Brazilian Symposium on Databases (SBBD 2003), pp. 304–318 (2003)

[10] Oliveira, S.R.M., Zaiane, O.R.: Toward standardization in privacy preserving data mining. In: ACM SIGKDD 3rd Workshop on Data Mining Standards, pp. 7–17 (2004).

ISSN: 2278 – 1323