Frequent Pattern Mining Algorithms Analysis

Ritesh Giri
Mumbai University
Rizvi Nagar,2-Dwing/301
Santacruz(West), Mumbai

Ananta Bhatt
Mumbai University
401, Royal Court ,Vijayanagar
Andheri(East), Mumbai

Aadhya Bhatt
Mumbai University
401, Royal Court ,Vijayanagar
Andheri(East), Mumbai

ABSTRACT
Frequent pattern mining is the most researched field in data mining. This paper provides comparative study of fundamental algorithms and performance analysis with respect to both execution time and memory usage. It also provides brief overview of current trends in frequent pattern mining and it applications. There are two categories of frequent pattern mining the algorithm, namely Apriori algorithm and Tree structure algorithm. The Apriori based algorithm uses generate and test strategy approach to find frequent pattern by constructing candidate items and checking their counts and frequency from transactional databases. The Tree structure algorithm uses a text only approach. There is no need to generate candidate item sets. Many tree based structures have been proposed to represent the data for efficient pattern discovery including FP-Tree, CAT-Tree, CAN-Tree, CP-Tree, and etc. Most of the tree based structure allows efficient mining with single scan over the database. In this paper, we describe the formatting guidelines for IJCA Journal Submission.

General Terms
Your general terms must be any term which can be used for general classification of the submitted material such as Pattern Recognition, Security, Algorithms et. al.

Keywords
Frequent Pattern, Data mining, Apriori, ECLAT, RElim, SaM, FP-Tree, CATS-Tree, CAN-Tree, CP-Tree.

1. INTRODUCTION
Recently there is a huge growth in the size of database which has led to a growing interest in the development of tool capable in the automatic extraction of knowledge from data. The term data mining or knowledge discovery in database has been adopted for a field of research dealing with the automatic discovery of implicit information or knowledge within the databases. The implicit information within databases, mainly the interesting association relationships among set of objects that led to association rules may disclose useful pattern for decision support, financial forecast, marketing policies, even medical diagnosis and many other applications. Data mining is the way of finding co-relation or patterns among dozens of fields in large relational databases. The automated, prospective analysis offered by data mining move beyond the analyses of past events provided by retrospective tools typical of decision support systems. Data mining is the tool that can answer business questions that traditionally were too tim consuming to resolve. We ask that authors follow some simple guidelines. In essence, we ask you to make your paper look exactly like this document. The easiest way to do this is simply to download the template, and replace the content with your own material.

2. RELATED WORK
Frequent pattern is defined as a pattern (A set of items, subsequences, substructures etc) that occurs frequently in a dataset. In association rule mining finding frequent patterns from databases is time consuming process. Several effective data structures such as two dimensional arrays, graphs, trees and tries have been proposed to collect candidate item sets and frequent item sets. But out of which the tree structure is most extractive to storing item sets.

A number of research works have been published that presenting new algorithm or improvements on existing algorithms to solve data mining problem efficiently. In that Apriori algorithm is the first algorithm proposed in this field. By the time of change or improvement in Apriori algorithm, the algorithms that compressed large database into small tree data structure like FP-Tree, CAT-Tree, CAN-Tree, CP-Tree have been discovered, These algorithms are partitioned based, divide and conquer method used that decompose mining task into smaller set of task for mining confined patterns in conditional database, which dramatically reduce search space. Hence there is a need of develop such a data structure which construct compact prefix free structure from one database scan and it provide same mining performance as FP- growth technique by efficient tree restructuring process. It should also support interactive and incremental mining without rescanning the original database.

2.1 Application of Frequent pattern Mining
Frequent patterns reflecting strong associations among multiple items or object, capture the underlying semantics in data. They were successfully applied to inter-disciplinary domains beyond data mining. Frequent pattern mining has huge application such as :

- Indexing and similarity search of complex structured data
- Spatiotemporal and multimedia datamining
- Stream data mining
- Web mining
- Software Bug mining and page-fetch

3. DATA MINING ALGORITHMS
Comparative study includes depth analysis algorithms and discusses some problems of generating frequent item sets from the algorithm. The comparative study of algorithm includes aspects like different support values, size of transactions and different data sets. One major function of association rules is to analyze large amounts of market basket transaction. Association rules have been applied to many areas including outlier detection, classification, clustering etc. The mining process can be broken down into the frequent item sets and the generation association rules.

3.1 Frequent Item Set Mining Methods

- Apriori: A candidate generation and test approach
- Improving the efficiency of Apriori algorithm
- FP-Growth: without candidate generation approach
- ECLAT: Frequent pattern mining with vertical data format
- Mining close frequent patterns and max-patterns.

3.2 Apriori Algorithm

The first algorithm used to determine the frequent item sets and to generate the Boolean association rules was the AIS algorithm introduces by Agrawal and Srikant. The Apriori algorithm introduced by the same author adds a major improvement to the history of determining the association rules. The Apriori algorithm tries to reduce the high number of database scans in order to determine the support , by significantly reducing the number of candidate item set. The basis for this reduction is the following property (Apriori property).

3.2.1 Performance

The Apriori algorithm is an important algorithm for historical reasons and also because it is simple algorithm that is easy to learn. However, faster and more memory efficient algorithms have been proposed. If efficiency is required, it is recommended to use a more efficient algorithm like FP-Growth

3.3 FP-Growth Algorithm

FP-Growth is an algorithm for discovering frequent item sets in a transaction database. It was proposed by Han et al.(2000). FP-Growth is a very fast and memory efficient algorithm. It uses a special internal structure called an FP-Tree. It allows frequent item set discovery without candidate item set generation, it is a two-step approach

1. Build a compact data structure called FP-Tree.
2. Extracts frequent item sets from the FP-Tree

   FP-Tree is constructed by using two passes over data set.
3. Pass1
   - Scan the data and find support for each item.
   - Discard infrequent items
   - Sort frequent items in decreasing order based on their support
4. Pass2
   - Construct the FP Tree by reading the transactions.

3.3.1 Performance

FP-Growth is generally the fastest and the most memory efficient algorithm. FP-Growth algorithm is efficient and scalable for mining both long and short frequent patterns. It is faster than the Apriori algorithm.

3.4 Eclat Algorithm

Eclat (equivalence class transformation) is an algorithm for discovering frequent item sets in a transaction database. It was proposed by Zaki in 2001. Contrarily to algorithm such as Apriori, Eclat uses a depth first search for discovering frequent item sets instead of a breadth-first search. Eclat algorithm is basically a depth first search algorithm using set intersection. It uses a vertical database layout i.e. instead of explicitly listing all transactions; each item is stored together with its cover (also called tid-list) and uses the intersection based approach to compute the support of an item set. In this way, the support of an item set X can be easily computed by simply intersecting the covers of any two subsets Y,Z,Y U Z=X. It states that, when the database is stored in the vertical layout, the support of a set can be counted much easier by imply intersecting the covers of two of its subset that together give the set itself.

3.4.1 Advantages

- There is no need to scan the database to find the support of k+1 item set. This is because of TID set of each k- item set carries the complete information required for counting such support.
- It is possible to significantly reduce this total size by generating collections of candidate item sets in a depth first strategy.
- This is always not possible since the total size of all covers at a certain iteration of the local item set generation procedure could exceed main memory limits.

3.4.2 Disadvantages

- Its fails to manage main memory at time of high candidate item sets.
- The merge routine contains a large amount of conditional branches, which are extremely badly predictable.

3.4.3 Performance

EClat is one of the interesting algorithm because it uses a depth-first search. In case of mining high utility item sets, the search procedure of EClat works very well.

3.5 Relim Algorithm

Relim (Recursive Elimination) is an algorithm for discovering frequent item sets in a transaction database. RElim was proposed by Borgelt (2005). RElim algorithm employs a basically horizontal transaction representation, but separates the transaction (or transaction suffixes) according to their leading items, thus introducing vertical representation aspect. The transaction database to mine is preprocessed. It does its work without prefix trees or another complicated data structures, processing the transactions directly. Due to simple structure of Relim; all the work is done in one simple recursive function, which can be written with relatively few lines of code.

3.5.1 Performance

RElim is based on deleting items, recursive processing, and reassigning transactions. It is very simple and works without complicated data structures. If a quick and straight forward implementation is desired, it could be the method of choice.

3.6 SaM Algorithm

The SaM (Split and Merge) algorithm established by Christian Borgelt. It is simplification of the already fairly simple RElim algorithm. While RElim represents a (conditional) database by storing one transaction list for each item (partially vertical representation), the split and merge algorithm employs only a

single transaction list (purely horizontal representation), stored as an array. This array is processed with a simple split and merge scheme, which computes a conditional database, processes this conditional database recursively and finally eliminates the split item from the original (conditional) database.

3.6.1 Advantages

- Simple data structure and processing scheme, easy to implement.
- Convenient to execute on external storage, thus rendering it a highly useful method if the transaction database to mine cannot be loaded into main memory.

3.6.2 Performance

SaM (Split and Merge) algorithm is an improved version of the Relim algorithm, both of which distinguish themselves from the other algorithms for frequent item sets mining by their simple processing scheme and data structure. SaM algorithm is well suited for an implementation that works on external storage, since it employs a simple array that can easily be represented a s a table in a relational database system.

4. DATA STRUCTURES USED

4.1 FP-Tree

To improve the efficiency of mining process, Han et. Al proposed an alternative framework, namely a tree based on the framework which is used to construct an extended prefix-tree-structure, called frequent pattern tree (FR-Tree) to capture the content of the transactional database rather than employing the generate-and-test strategy of Apriori algorithm, such tree based algorithm focus on frequent pattern growth-which is restricted test-only approach (i.e. doesn't generate candidate, and only test for the frequency). Improvement in Apriori algorithm, which compressed a large database into small tree data structure like FP-Tree. Times Roman font, or other Roman font with serifs, as close as possible in

4.2 CATS-Tree

CATS Tree (compressed and arranged transaction sequences) extends the idea of FP-Tree to improve storage compression and allow frequent pattern mining without generation of candidate item set. It allows mining only through a single pass over the database. CATS – Tree has several common properties of FP-Tree.

4.3 CAN-Tree

CAN-Tree (Canonical-order tree), that captures the complete information in a canonical order of terms from database into a prefix-tree structure in order to facilitate for incremental and interactive mining using FP-growth mining technique. CAN-Tree capture he contents of the transaction database and orders tree nodes according to some canonical order. The construction of the CAN Tree only requires one database scan as compared to an FP-Tree which require two database scans.

4.4 CP-Tree

CP-Tree (Compact pattern tree) captures database information with one scan (insertion phase) and provides the performance as the FP-growth method (restructuring phase) by dynamic tree restructuring process. Moreover, CP-Tree can give full functionality for interactive and incremental mining. CP-Tree is efficient for frequent pattern mining, interactive, and incremental mining with single database scan. Roman in which these guidelines have been set. The goal is to have a 9-point text, as you see here. Please use sans-serif or non-proportional fonts only for special purposes, such as distinguishing source code.

5. RESULT

Experiments are performed on the five datasets namely adult, census, accident, retail, and mushroom. Machine with configuration of windows Vista operating system and 2-GB of RAM is used. The results were compared to experiments with Borgelt's implementations of APriori, FP-Growth, EClat, RELim and SaM. All experiments were re-run to ensure that the results are comparable.

Table 1. Dataset description

Database	#Itemset	Avg. Length	#Transactions
Adult	48842	3.79	45223
Census	48842	49.5	199523
Mushroom	119	23	8124
Retail	16469	10.3	88162
Accident	468	33.8	340183

Table 2. Execution of Adult Dataset

Support	Apriori	FP-Growth	Eclat	Relim	SaM
30	0.58	0.56	0.54	0.49	0.47
40	0.54	0.50	0.49	0.44	0.44
50	0.50	0.49	0.45	0.42	0.41
60	0.48	0.48	0.44	0.40	0.40
70	0.45	0.42	0.40	0.39	0.37

Table 3. Execution of Census Dataset

Support	Apriori	FP-Growth	Eclat	Relim	SaM
30	0.87	1.21	0.76	0.74	0.74
40	0.86	1.16	0.75	0.72	0.71
50	0.79	0.86	0.72	0.70	0.69
60	0.75	0.73	0.69	0.67	0.67
70	0.73	0.68	0.65	0.65	0.64

Table 4. Execution of Mushroom Dataset

Support	Apriori	FP-Growth	Eclat	Relim	SaM
30	0.14	0.13	0.11	0.11	0.11
40	0.13	0.11	0.11	0.09	0.11
50	0.17	0.19	0.09	0.09	0.08
60	0.10	0.08	0.09	0.08	0.07
70	0.09	0.08	0.08	0.07	0.06

Table 5. Execution of Retail Dataset

Support	Apriori	FP-Growth	Eclat	Relim	SaM
30	0.88	0.89	0.86	0.85	0.85
40	0.85	0.87	0.85	0.84	0.83
50	0.81	0.82	0.80	0.78	0.78
60	0.79	0.78	0.76	0.74	0.73
70	0.74	0.72	0.68	0.66	0.65

Table 6. Execution of Accident Dataset

Support	Apriori	FP-Growth	Eclat	Relim	SaM
30	1.22	1.16	1.14	1.14	1.12
40	1.18	1.13	1.10	1.10	1.07
50	1.10	1.08	1.06	1.04	1.02
60	1.05	1.02	0.98	0.96	0.94
70	0.98	0.95	0.93	0.93	0.91

6. CONCLUSION

The study is carried out to analyze the performance of different frequent pattern mining algorithms when applied to real world datasets. After obtaining results, it is observed that behavior of frequent pattern mining algorithms varies differently for each dataset. There is need to propose a novel tree data structure to extract all frequent patterns from transactional database with single database scan and without rescan the original database. A new tree data structure should save search space by storing only frequent items. It should support interactive mining like CAN-Tree and CP-Tree, means if users specified minimum support is changed then also it can extract frequent patterns without the need to rescan the database. It should also support incremental mining like CAN-Tree and CP-Tree, means later if transaction is added or old transactions are deleted then also it can extract frequent patterns without the need to rescan the original database.

7. REFERENCES

[1] Syed Khairuzzaman Tanbeer, Chowdhary Farhan Ahmed, Byeong-Soo Jeong and Y.W.Lee, CP-Tree: A Tree Structure for Single-Pass Frequent Pattern Mining ,In proceedings of 12th Pacific Asia Conference,2008.

[2] C. Borgelt. An Implementation of the FP- growth Algorithm. Proc. Workshop Open Software for Data Mining (OSDM'05 at KDD'05, Chicago,IL),1–5.ACMPress, New York, NY, USA 2005.