International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 4, April - 2014

A Survey on Frequent Pattern Mining Algorithms

Sumit Aggarwal
Research Scholar
Department of Computer Science and Application
Kurukshetra University, Kurukshetra (India)

Vinay Singal
Research Scholar
Computer Engineering, U.I.E.T.
Kurukshetra University, Kurukshetra(India)

Abstract: ***Web mining can simply be described as the application of data mining techniques to the Web. It can be broadly divided in to three categories: Web content mining, Web structure mining and Web usage mining. We will be discussing the web usage mining techniques in this text. In web usage mining association rules are generated from the frequent patterns. The two widely used frequent pattern mining algorithms are apriori algorithm and FP-growth algorithm.***

Keywords: Apriori Algorithm, FP-growth Algorithm, FP-tree, Web Usage Mining.

I. INTRODUCTION

World Wide Web is a huge, dynamic and unstructured data repository. The use of WWW has increased tremendously in the near past and the information available on internet has had an explosive growth. It has become very difficult to access relative information on the web. This has lead to a hot research in the field of web usage mining. Web usage mining helps to understand the behavior of the customer and to evaluate the efficiency and performance of particular web site [4].

The focus of this paper is to provide an overview on the two main Frequent Pattern Mining (FPM) algorithms: Apriori algorithm and FP-growth algorithm. The rest of the text is organized as follows: Section II provides an overview of the web usage mining. In section III, we have discussed the problem statement and some terminology related to the research is defined. Section IV, contains two FPM algorithms which are the main part of this text. Section V will give a comparison of the algorithms which are studied and finally section VI, contain the conclusion of the text.

II. WEB USAGE MINING

Web usage mining refers to the discovery of interesting association rules from the data generated as a result of interaction of users with the various Web resources [2]. The association rules will represent the information regarding the resources and pages that are frequently accessed by various user groups having common needs and interests.

The main goal of the web usage mining is to collect, model and analyze the behavioral patterns of the users interacting with a web site. The data used for mining can be collected at the servers, clients or the proxy servers. In web usage mining the data consists of pattern of usage of the resources. It is represented using web log files. These log files record each page request information.

The most significant phase of the web usage mining is the discovery of frequent patterns from the web log files using FPM algorithms. Using these frequent patterns association rules are generated, which is a straightforward task because we don't need to refer the databases.

III. PROBLEM STATEMENT

In web usage mining, association rules are generated using frequent patterns. The problem consists of two phases [5]:

1. Firstly we find the frequent item sets. They are the set of items whose occurrence exceeds a predefined threshold value.
2. Then we generate the association rules from the frequent item sets, with a constraint of minimum confidence.

A more formal definition of the problem is as follows [1]:
Let I: $\{i_1, i_2, i_3, \ldots\ldots i_n\}$ be a set of distinct items. D be a database containing transaction over I in which each transaction consists of a set of items $i_1, i_2, i_3, \ldots i_n \subseteq I$. Each transaction in the database is associated with an transaction identifier TID. We say that the transaction T contain X if $X \subseteq T$. Here X is a set of some items in I. The association rules are the implications of the form $X \Rightarrow Y$. Where $X \subset I$, $Y \subset I$ and $X \cap Y = \varnothing$. Here X is called the antecedent and Y is called the consequent of the rule. The rule $X \Rightarrow Y$ holds in database D with confidence c if c% of the transactions in D containing X also contains Y in them. The rule $X \Rightarrow Y$ has a support s in the database D if s% of transactions in D contains $X \cup Y$.
Confidence and support are significant measures of rule interestingness and they reflect usefulness and certainty of rule respectively [5]. The selection of the association rules depend upon these two values because the selected rules should have support and confidence greater than the respective threshold values.

$$\text{Support } (X \Rightarrow Y) = \frac{\text{Frequency } (X \cup Y)}{/D/}$$

$$\text{Confidence } (X \Rightarrow Y) = \frac{\text{Frequency } (X \cup Y)}{\text{Frequency } (X)}$$

A set of items is called itemset. And the itemset having k items is called k-itemset. The support count of itemset gives us the number of transactions in the database containing the itemset. The number of transactions that should contain the itemset to fulfill the minimum support condition is called the minimum support count [1].

The definition of frequent item sets can be given as, the itemset which has a support count greater than the minimum support count.

IV. FREQUENT PATTERN MINING ALGORITHMS

A. Apriori Algorithm:

Apriori algorithm was proposed by R. Aggarwal and R. Srikant in 1994 for finding the interesting association rules from the databases. Figure 1 gives the apriori algorithm.

The algorithm [1] contains two functions apriori_gen() and subset().

The apriori_gen() takes L_{k-1} as argument, which represents the set of large (k-1)-itemsets. The function returns the superset of the set of all large k-itemsets.

L_k	Set of large k-itemsets having minimum support. Each element of this set has two values: Itemset and Support_count.
C_k	Set of candidate k-itemsets. Each of the element has two values: Itemset and Support_count.

Table 1[1]

The table above represents the notation used.

1) L_1={ Large 1-itemsets};
2) For(k=2; L_{k-1}!= ∅; k++) do begin
3) C_k= apriori_gen(L_{k-1}); //New candidate
4) For all transactions t∈D do begin
5) C_t= Subset(C_k,t) //Candidates in t
6) For all candidates c∈C_t do
7) c.count ++;
8) End
9) L_k= {c∈ C_t| c.count >= minsup}
10) End
11) Answer= $\cup_k L_k$

Figure 1[1]

The subset () function takes two arguments, C_k and t. Here t is the transaction and C_k represents the set of candidate k-itemsets. This function gives us all the candidates present in transaction t. The working of the algorithm is explained as under:

The first pass of algorithm will give us the large 1-itemsets by counting occurrences of items. Any subsequent pass k of the algorithm will consist of two phases. In first phase, apriori_gen function is run to generate the candidate itemsets C_k using the large itemsets L_{k-1} generated in the k-1 pass. In second phase, the support of the candidates in C_k is counted using database scan. For determining the candidates in C_k present in any transactions t, subset() function is used as described above.

B. FP-growth Algorithm:

The Apriori algorithm generated a large number of candidate sets. A new method, FP-growth, was purposed for frequent itemsets generation without candidate itemset generation. This method involves two phases. First phase consist of constructing a compact data structure called FP-tree [3]. In second phase, we extract frequent itemsets using FP-growth algorithm over FP-tree generated in previous phase.

First phase need two database scans for generating the FP-tree. But the second phase don't need any scan over database and it uses on FP-tree to generate frequent itemset.

FP-tree can be defined as follows [3]:

1. It consists of a root labeled null, a frequent item header table and a set of item prefix subtrees as the children of root.

2. Each node consists of: item-name, count and node-link. Item-name gives us the name of item represented by the node, count will give us the number of transactions that can be represented by the path leading to the node, node-link contains the pointer to the next node containing the same item or null if there is none.

3. Frequent item header table consists of: item-name and head of node-link which have a pointer to the first node in the tree containing the item-name.

The two phases of FP-growth can be represented by two algorithms as described below:

Algorithm 1[3]: (Phase 1: FP-tree const.)

Input: A transactional database DB and a minimum support threshold min_sup.

Output: Its frequent pattern tree, FP-tree

Method: The FP-tree is constructed in following steps:

1. Scan the transaction database *D* once. Collect the set of frequent items *F* and their support. Sort *F* in support descending order as *L*, the *list* of frequent items.

2. Create the root of an FP-tree, *FPT*, and label it as "root". For each transaction *T* in *D* do the following:

a. Select and sort the frequent items in *T* according to the order *L*. Let the sorted frequent item list in *T* be [*p*|*P*], where *p* is the first element and *P* is the remaining list. Call *insert_tree*([*p*|*P*],*FPT*).

b. The function *insert_tree*([*p*|*P*],*FPT*) is performed as follows. If *FPT* has a child *N* such that *N.item-name=p.item-name*, then increment *N*'s count by 1; else create a new node *N*, and let its count be 1, its parent link be linked to *FPT*, and its node-link be linked to the nodes with the same *item-name* via the node-link structure. If *P* is nonempty, call *insert_tree(P,N)* recursively.

Algorithm 2[3]: (Phase 2: FP-growth)

Input: A *FP*-tree constructed with above algorithm;
D-transaction database;
s- Minimum support threshold.

Output: The complete set of frequent patterns.

Method:
Call FP-growth (*FP-tree, null*).
Procedure FP-growth (*Tree, A*)
{
1.) If *Tree* contains a single path *P*
2.) Then for each combination (denoted as *B*) of the nodes in the path *P* do
3.) Generate pattern $B \cup A$ with *support = minimum support of node in B*;
4.) Else for each a_i in the header of the *Tree* do{
5.) Generate pattern $B = a_i \cup A$ with support = a_i.*support*;
6.) Construct *B*'s conditional pattern base and *B*'s conditional FP-tree $Tree_B$;
7.) If $Tree_B \neq \emptyset$
8.) Then call FP-growth ($Tree_B$, *B*)}
}

V. COMPARISON OF THE ALGORITHMS

The two algorithms discussed above are widely studied algorithms for frequent pattern mining. The apriori algorithm works by generating candidate itemsets while the FP-growth algorithm works without generating the candidate sets.

The apriori algorithm has the following bottlenecks:
1.) Difficult to handle huge number of candidate itemsets. The candidate generation can be very costly with the increasing size of database.
2.) It is tedious to repeatedly scan the huge databases.

The FP-growth algorithm is quite a different algorithm from its predecessors. It works by generating a prefix-tree data structure known as FP-tree from two scans of the database. This algorithm doesn't need to scan the database multiple times. The main drawbacks of the apriori algorithm are removed with the introduction of the FP-growth algorithm. The table represents the comparison of two algorithms studied here based on different parameters.

Parameter	Apriori Algorithm	FP-Growth algorithm
Technique	Use Apriori property and join and prune property	Constructs FP-tree and conditional pattern base satisfying minimum support.
Memory Utilization	Large memory space for candidate itemsets	Lesser memory due to compact structure
No, of scans	Multiple scans of database.	Scans the database twice only
Time	Execution time is large because of candidate itemsets generation.	Execution time is smaller.

Table 2: Comparison of Algorithms

VI. CONCLUSION

It is evident from the study that FP-growth is a far better algorithm than the apriori algorithm. It can also be shown from the experimental data that in the apriori algorithm the performance is influenced by the support factor. And the number of database scans in apriori algorithm increases with the dimensions of the candidate itemsets. So the apriori algorithm works well only with the small databases and with large support factor. On the other hand, FP-growth algorithm works very well with large databases as there are only two database scans and no candidate generation. But some future work can be done to increase the efficiency of the apriori algorithm. In future these algorithms can also be extended to web content mining and web structure mining.

VII. REFERENCES

[1] Agrawal R., Srikant R., "*Fast Algorithm for Mining Association Rules*", VLDB. Sep 12-15 1994, Chile, 487-99, pdf, ISBN 1-55860-153-8.

[2] Huiping Peng, "Discovery of Interesting Association Rules Based on Web Usage Mining" 2010 International Conference.

[3] Han J., Pei J., Yin Y., Mao R., "Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach" Data Mining and Knowledge Discovery, 2004, 53-87.

[4] K. R. Suneetha and Dr. R. Krishnamoorthi "Identifying User Behavior by Analyzing Web Server Access Log File", International Journal of Computer Science and Network Security, vol. 9, no. 4, 2009.

[5] Goswami D.N., Chaturvedi Anshu., Raghuvanshi C.S., "An Algorithm for Frequent Pattern Mining Based on Apriori", IJCSE, Vol. 02, No. 04, 2010, 942-947.