Mining Maximal Frequent Item Sets

Dr. S.S. Mantha
Chairman AICTE
Professor.
CAD/CAM Robotics
VJTI
Mumbai, India

Madhuri Rao
M. E .Computer
Engineering
Assistant Professor
TSEC, Bandra (w)
Mumbai-50,India

Ashwini Anil Mane
B. E. Computer
Engineering
Lecturer
TSEC, Bandra (w)
Mumbai-50,India

Anil S. Mane
B.E. EXTC Engineering
Team Leader
Patni Computers Systems
Ltd, Airoli
Navi Mumbai-400708,
India

ABSTRACT

Data mining or knowledge discovery in databases (KDD) is a collection of exploration techniques based on advanced analytical methods and tools for handling a large amount of information. Mining association rule is a main content of data mining research at present, and emphasizes particularly is finding the relation of different items in the database. How to generate frequent item sets is the key and core. It is an important aspect in improving mining algorithm that how to decrease item set candidates in order to generate frequent item set effectively.

Efficient algorithms for mining frequent items etc are crucial for mining association rules. Most existing work focuses on mining all frequent item sets (FI). However, since any subset of a frequent item set also is frequent, it is sufficient to mine only the set of maximal frequent item sets (MFI). In this paper we study the performance of existing approach, Max-Miner, for mining maximal frequent item sets. We have also developed an algorithm, called M-fp. We also present experimental results which shows that our method outperforms the existing method Max-Miner.

Keywords

Frequent item sets, closed frequent item sets, Maximal frequent item sets, Association rules.

1. INTRODUCTION

Let $I=\{i1, i2, \ldots, in\}$ be a set of items , We call $X \subseteq I$ an item set, and we call X a k-item set if the cardinality of item set X is k. Let database T be a multi set of subsets of I, and let $support(X)$ be the percentage of item set Y in T such that $X \subseteq Y$. Informally, the support of an item set measures how often X occurs in the database. If $support(X) \geq minSup$, we say that X is a frequent item set , and we denote the set of all frequent item sets by FI.

A closed frequent item set is a frequent item set X such that there exists no superset of X with the same support count as X. If X is frequent and no superset of X is frequent, we say that X is a maximal frequent item set, and we denote the set of all maximal frequent item sets by MFI.

There are basically two types of algorithms to mine frequent item sets, breadth first algorithms and depth first algorithms. The breadth first algorithms, such as Apriori [1], apply a bottom-up level-wise search in the item set lattice. Candidate item sets with k+1 items are only generated from frequent item sets with k items. For each level, all candidate item sets are tested for frequency by scanning the database. On the other hand, depth first algorithms such as FP-growth [2] search the lattice bottom-up in depth first way. From a singleton item set $\{i\}$, successively larger candidate sets are generated by adding one element at a time

The drawback of mining all frequent item sets is that if there is a large frequent item set with size l, then almost all 2^l candidate subsets of the item set might be generated. However, since frequent item sets are upward closed, it is sufficient to discover only all maximal frequent item sets (MFI) .

Bayardo [3] introduces Max-Miner which extends Apriori to mine only"long" patterns (maximal frequent item sets). To reduce the search space, Max-Miner performs not only subset infrequency pruning such that a candidate item set that has an infrequent subset will not be considered, but also a "look ahead" to do superset frequency pruning. Though superset frequency pruning reduces the search time dramatically, Max-Miner still needs many passes to get all long patterns.

In [4], Burdick, Calimlim, and Gehrke gave an algorithm called MAFIA to mine maximal frequent item sets. This method uses a bitmap representation, where the count of an item set is based on the column in the bitmap (the bitmap is called"vertical bitmap"). As an example, the bit vectors for items B, C, and D are 111110, 011111, and 110110, respectively. To get the bit vectors for any item set, we only need to apply the bit vector and operation $\otimes$ on the bit vectors of the items in the item set. For above example, the bit vector for item set BC is $111110 \otimes 011111$, which equals 011110, while the bitmap for item set BCD can be calculated from the bitmaps of BC and D, i.e., $011110 \otimes 110110$, which is 010110. The count of an item set is the number of 1's in its bitvector. A new strategy is used, called PEP. MAFIA is a depth first algorithm.

G¨osta Grahne and Jianfei Zhu [5] introduces FpMAX, which an extension of the FP-growth method, for mining MFI only. During the mining process, an FP-tree (Frequent Pattern tree) is used to store the frequency information of the whole dataset. To test if a frequent item set is maximal, another structure called a Maximal Frequent Item set tree (MFI-tree) is utilized to keep track of all maximal frequent item sets. This structure makes Fp-MAX effectively reduce the search time and the number of subset testing operations.

2. RELATED THEORY

2.1. FP-tree and FP-growth method

In the aforementioned FP-growth method [2], a novel data structure, the FP-tree (Frequent Pattern tree) is used. The FP-tree is a compact data structure for storing all necessary information about frequent item sets in a database. Every branch of the FP-tree represents a frequent item set, and the nodes along the branch are ordered decreasingly by the frequency of the corresponding item, with leaves representing the least frequent items. Each node in the FP-tree has three fields: *item-name*, *count* and *node-link*, when *item-name* registers which item this node represents, *count* registers the number of transactions represented by the portion for the path reaching this node, and *node-link* links to the next node in the FP-tree carrying the same *item-name*, or null if there is none. The FP-tree has a header table associated with it. Single items are stored in the header table in decreasing order of frequency. Each entry in the header table consists of two fields, *item-name* and head of *node-link* (a pointer pointing to the first node in the FP-tree carrying the *item-name*).

Compared with Apriori [1] and its variants which need several database scans, the FP-growth method only needs two database scans when mining all frequent item sets. In the first scan, all frequent items are found. The second scan constructs the first FP-tree which contains all frequency information of the original dataset. Mining the database then becomes mining the FP-tree. Figure 1 shows a database example. After the first scan, all frequent items are inserted in the header table of an initial FP-tree. Figure 2 shows the first FP-tree constructed from the second scan. The FP-growth method relies on the following principle: if X and Y are two item sets, the support of item set $X \cup Y$ in the database is exactly that of Y in the restriction of the database to those transactions containing X. This restriction of the database is called the conditional pattern base of X. Given an item in the header table, the growth method constructs a new FP-tree corresponding to the frequency information in the sub-dataset of only those transactions that contain the given item. Figure 3 shows the conditional pattern base and the FP-tree for item *{p}*, this step is applied recursively, and it stops when the resulting smaller FP-tree contains only one single path. The complete set of frequent item sets is generated from all single path FP-trees. When adding an item i to the existing item set *head*, we denote the item set *head* $\cup$ i by Z, the path from the parent node of this node (node's *item-name* is i) to the root node in the *head*'s FP-tree is called Z's prefix path. Figure 4 shows the prefix paths for item *{p}*.

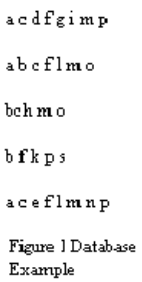

Figure 1 Database Example

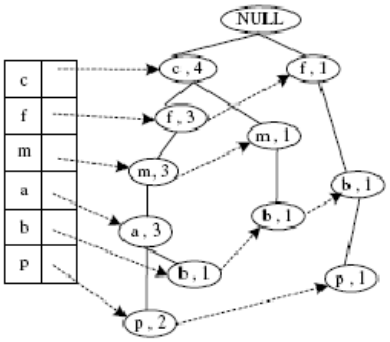

Figure .2 FP-tree

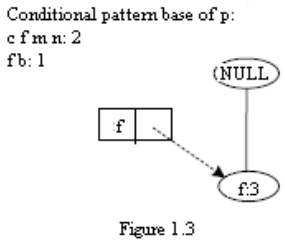

Figure 1.3

Figure .3 Conditional pattern base of p

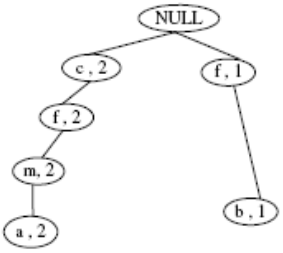

Figure 4 Prefix paths for item {p}

2.2. Max-Miner method

Bayardo proposed the concept of MFI (Maximal Frequent Item sets) and the Max-Miner algorithm for mining only MFI [3]. Max-Miner looks only the MFIs, because of that, the search space can be reduced. Max-Miner uses a bottom up traversal of a database. Max-Miner employs a purely breadth-first search of the set-enumeration tree in order to limit the number of passes made over the data.

The key to an efficient set-enumeration search is the pruning strategies that are applied to remove entire branches from consideration. Without pruning, a set-enumeration tree search for frequent item sets will consider every item set over the set of all items. Max-Miner uses pruning based on subset infrequency, as does Apriori [1], but it also uses pruning based on superset frequency. In Max-Miner each node represent in the set enumeration tree let us call it a candidate group. A candidate group g consists of two item sets. The first, called the head and denoted h (g), represents the item set enumerated by the node. The second item set, called the tail and denoted t (g), is an ordered set and contains all items not in h (g) that can potentially appear in any sub-node. The ordering of tail items reflect how the sub-nodes are to be expanded. In the case of a static lexical ordering without pruning, the tail of any candidate group is trivially the set of all items following the greatest item in the head according to the item ordering.

Count the support of a candidate group g, means compute the support of item sets h(g), h(g) U t(g), and h(g) U {i} for all i € t(g). The supports of item sets other than h (g) are used for pruning.

Candidate group

- Head: h(g)
  Item set enumerated by the node.
- Tail: t(g)
  An ordered set and contains all items not in h (g)

Steps to follow:

1. Count the support of a candidate group g, and also compute the support for h(g), h(g) U t(g) and h(g) U {i} for each i in t(g).
2. If h (g) U t (g) is frequent, then stop expanding the node g and report the union as frequent item set.

3. If h(g) U {i} is infrequent, then remove i from all sub nodes (just remove i from any tail of a group after g).
4. Expand the node g by one and repeat the same.

3 . MINING MFI BY M-fp

Figure 5 shows steps for finding the maximal frequent item sets using M-fp method

Procedure MFI (T)

Input: T: an FP-tree

Output: all Maximal Frequent Item sets.

Method:

Begin

Get the pointer p to the first node of the bottom layer tree

If the occurrence of p is greater than or equal to min_sup

Then

FI = FI U I_p

Else For each q in the right list of the same layer

Then

If the occurrence of p + q is greater than or equal to min_sup

FI =FI U I_q

End if

End if

MFI= FI- subsets

End

Figure 5 M-fp Algorithm

4. EXPERIMENTAL RESULTS

The experiments are conducted on a Pentium® Dual-core CPU E5400 @ 270 GHz with 1.96 GB of RAM running Microsoft Windows XP Professional. All code is compiled using Microsoft Visual C# 3.5.

The two algorithms are compared for datasets Market-basket-data, Stationary-data, Computer shopping data. Figure 6 and 7 illustrate the running time results of comparing M-fp, Max-Miner.

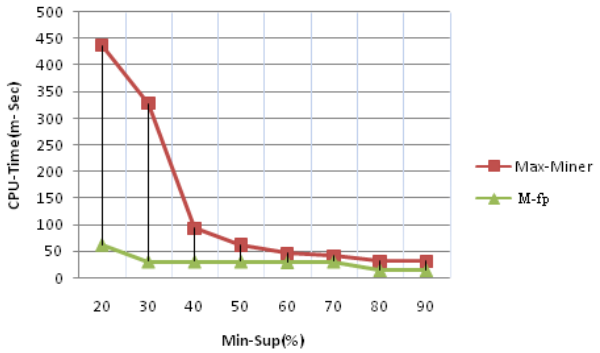

Figure 6 Dataset Market-basket-data

We can see that the M-fp method outperforms the Max-Miner method. In the graph shown in Figure 6 x-axis is the user-specified minimum support in percent, while the y-axis is the algorithms running time in milliseconds. Figure 6 shows the running time results of comparing M-fp, Max-Miner on Market-basket-data. Figure 7 shows the running time results of comparing M-fp, Max-Miner on shopping data.

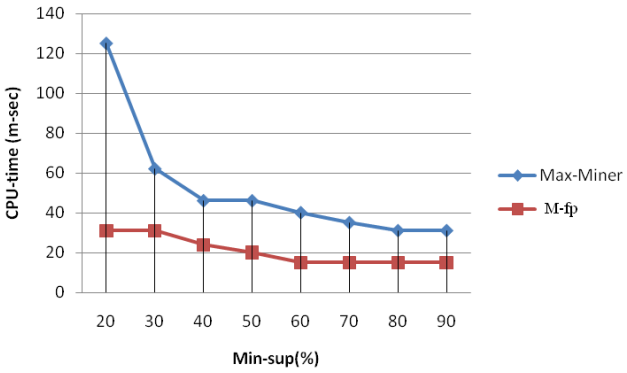

Figure 7 Dataset shopping data

Figure 8 shows the running time results of comparing M-fp, Max-Miner on Stationary-data.

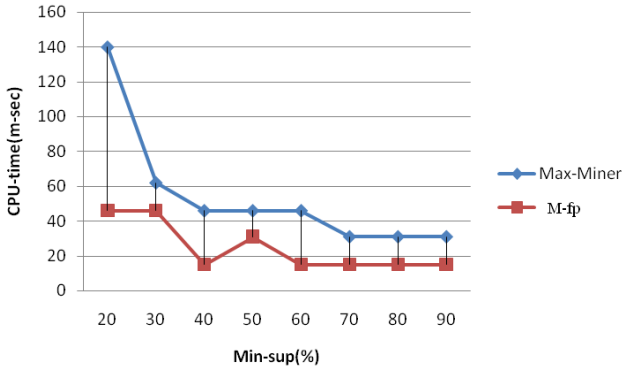

Figure 8 Computer Stationary-data

5. CONCLUSIONS

In this paper we present the two methods for finding Maximal frequent item sets, first is Max-Miner method and second is FP-tree method. The Max-Miner algorithm for mining maximal frequent item set (MFI) looks only for the MFI and because of that search space can be reduced.

The Max-Miner performs not only subset infrequency pruning, where a candidate item sets with an infrequent subset will not be considered, but also a "look ahead" to do superset frequency pruning. But Max-Miner needs several passes of the database to find the maximal frequent item sets.

For FP-tree and the M-fp algorithm method superset checking not needed. The FP-tree data structure is used for storing frequency information of the original database in a compressed form. The FP-tree method finds frequent item sets with two database scans. Candidate generation is not required for FP-tree method.

6. REFERENCES

[1] R. Agarwal, T. Imielinski, and A. Swami, "Mining Association rules between Sets of Items in Large Databases", Proc. ACM SIGMOD International Conference on Management of Data (SIGMOD'93), Washington, D.C.,USA, pp. 2007.

[2] G. Grahne, and J.F. Zhu, "Fast Algorithms for Frequent Item set Mining Using FP-Trees," IEEE Transactions on Knowledge and Data Engineering, pp. 1347-1362, Oct 2005.

[3] R.J. Bayardo, "Efficiently Mining Long Patterns from Databases," Proc. ACM-SIGMOD Int'l Conf. Management of Data, pp. 85-93, 1998.

[4] D. Burdick, M. Calimlim, J. Flannick, J. Gehrke, and T. Yiu, "MAFIA: a maximal frequent itemset algorithm," IEEE Transactions on Knowledge and Data Engineering, pp. 1490–1504, Nov 2005.

[5] G. Grahne and J.F. Zhu, "High Performance Mining of Maximal Frequent Item sets", Proc. SIAM Int'l Conf. High Performance Data Mining, pp. 135−143, 2003.

[6] Lisheng Ma, Huiwen Deng, "Fast Algorithm for mining Maximal frequent Item sets", First International symposium on Data, Privacy and E-Commerce, pp. 86– 91, Nov 2007.

[7] Bo Liu, Jiuhui Pan "Graph-based Algorithm For mining Maximal Frequent Item sets", Forth International conference on Fuzzy Systems and Knowledge discovery (FSKD2007).

[8] J. Han, J. Pei, and Y. Yin, "Mining Frequent Patterns without Candidate Generation," Proc. ACM-SIGMOD Int'l Conf. Management of Data, pp. 1-12, May 2000.

[9] G. Grahne and J.F. Zhu, "High Performance Mining of Maximal Frequent Item sets", Proc. SIAM Int'l Conf. High Performance Data Mining, pp. 135−143, 2003.