A Frame Work for Frequent Pattern Mining Using Dynamic Function

Sunil Joshi[1], R S Jadon[2] and R C Jain[3]

[1] Computer Applications Department, Samrat Ashok Technological Institute Vidisha, M.P. , India

[2] Computer Applications Department, Madhav Institute of Technology and Science Gwalior, M.P. , India

[3] Computer Applications Department, Samrat Ashok Technological Institute Vidisha, M.P. , India

Abstract

Discovering frequent objects (item sets, sequential patterns) is one of the most vital fields in data mining. It is well understood that it require running time and memory for defining candidates and this is the motivation for developing large number of algorithm. Frequent patterns mining is the paying attention research issue in association rules analysis. Apriori algorithm is a standard algorithm of association rules mining. Plenty of algorithms for mining association rules and their mutations are projected on the foundation of Apriori Algorithm. Most of the earlier studies adopted Apriori-like algorithms which are based on generate-and-test candidates theme and improving algorithm approach and formation but no one give attention to the structure of database. Several modifications on apriori algorithms are focused on algorithm Strategy but no one-algorithm emphasis on least transaction and more attribute representation of database. We presented a new research trend on frequent pattern mining in which generate Transaction pair to lighten current methods from the traditional blockage, providing scalability to massive data sets and improving response time. In order to mine patterns in database with more columns than rows, we proposed a complete framework for the frequent pattern mining. A simple approach is if we generate pair of transaction instead of item id where attributes are much larger then transaction so result is very fast. Newly, different works anticipated a new way to mine patterns in transposed databases where there is a database with thousands of attributes but merely tens of stuff. We suggest a novel dynamic algorithm for frequent pattern mining in which generate transaction pair and for generating frequent pattern we find out by longest common subsequence using dynamic function. Our solutions give result more rapidly. A quantitative investigation of these tradeoffs is conducted through a wide investigational study on artificial and real-life data sets.

Keywords: *Longest Common Subsequence, Frequent Pattern mining, dynamic function, candidate, transaction pair, association rule, vertical mining*

1. Introduction

Frequent Pattern Mining is most dominant problem in association mining. Plenty of algorithms for mining association rules and their mutations are projected on the foundation of Apriori Algorithm. Most of the earlier studies adopted Apriori-like algorithms which are based on generate-and-test candidates theme and improving algorithm approach and formation but always focus on item id instead of transaction id. Several modifications on apriori algorithm are focused on algorithm Strategy but no one-algorithm emphasis on least transaction more attribute representation of database.

Most of the preceding work on mining frequent patterns is based on the horizontal representation. However, recently a number of vertical mining algorithms have been projected for mining frequent itemsets. Mining algorithms using the vertical representation have shown to be effective and usually do better than horizontal approaches [11]. This benefit stems from the fact that frequent patterns can be counted via tidset intersections, instead of using complex interior data structures like the hash/search trees that the horizontal algorithms need [10]. Also in the vertical mining, the candidate creation and counting phases are done in a single

step. This is done because vertical mining offers usual pruning of unrelated transactions as a result of an intersection. Another characteristic of vertical mining is the utilization of the autonomy of classes, where each frequent item is a class that contains a set of frequent k-itemsets (where k > 1) [6]. The vertical arrangement appears to be a usual choice for achieving association rule mining's purpose of discovering associated items. Computing the supports of itemsets is simpler and quicker with the vertical arrangement since it involves only the intersections of tid-lists or tid-vectors, operations that are well-supported by the current database systems. In difference, complex hash-tree data structures and functions are required to perform the same function for flat layouts. There is an automatic reduction of the database before each scan for those itemsets that are significant to the following scan of the mining process are accessed from disk. In the horizontal arrangement, however, irrelevant information that happens to be part of a row in which useful information is present is also transferred from disk to memory. This is because database reductions are moderately hard to implement in the horizontal arrangement. Further, still if reductions were possible, the irrelevant information can be removed only in the scan following the one in which its irrelevance is exposed. Therefore, there is always a reduction delay of at least one scan in the horizontal layout.

A simple approach is if we generate pair of transaction instead of item id where attributes are much larger then transaction then result is very fast. Recently, different works proposed a new way to mine patterns in transposed databases where a database with thousands of attributes but only tens of objects [15]. In this case, mining the transaction pair runs through a smaller search space. None algorithm filters or reduces the database in each pass of apriori algorithm to count the support of prune pattern candidate from database.

Most of the preceding work on vertical mining concentrates on intersection of transaction [12]. This is based on intersection of perpendicular tid-vector where it is a set of columns with each column storing an IID and a bit-vector of 1's and 0' to represent the occurrence or nonexistence, respectively, of the item in the set of customer transactions. If we use list-based layout then it takes much less space than the bit-vector approach (which has the overhead of openly representing absence) in sparse databases. We make the case in this paper and use list-based layout [16]. To find intersection we use dynamic technique instead of traditional approach. We suggest a novel dynamic algorithm for frequent pattern mining in which we generate transaction pair and for generating frequent pattern we find out by longest common subsequence using dynamic function.

The rest of this paper is structured as follows. Section II introduces the problem and reviews some efficient related works. The projected method is described in section III. Section IV explains in details the projected FPMDF algorithm. A justification with Example is given in Section V. The investigational results and assessment show in section VI. Finally Section VII contains the conclusions and upcoming works

2. Frequent Pattern Mining

Frequent Itemset Mining came from efforts to determine valuable patterns in customers' transaction databases. A customers' transaction database is a series of transactions (T = t1. . . tn), where each transaction is an itemset (ti $\subseteq$ I). An itemset with k elements is known as k-itemset. In the rest of the paper we make the (practical) assumption that the items are from a prearranged set, and transactions are stored as sorted itemsets. The support of an itemset X in T, denoted as suppT(X), is the number of those transactions that hold X, i.e. suppT(X) = |{tj : X $\subseteq$ tj}|. An itemset is frequent if its support is larger than a support threshold, originally denoted by min supp. The frequent itemset mining problem is to discover all frequent itemset in a given transaction database.

The primary Algorithm Proposed for finding frequent itemsets, is the APRIORI Algorithm [1]. This algorithm was enhanced later to obtain the frequent pattern quickly [2]. The Apriori algorithm employs the downhill closure property—if an itemset is not frequent, any superset of it cannot be frequent either. The Apriori Algorithm performs a breadth-first search in the search Space by generating candidate k+1 itemsets from frequent k-itemsets. The occurrence of an itemset is computed by counting its happening in each transaction. Numerous variants of the Apriori algorithm have been developed, like AprioriTid, AprioriHybrid, direct hashing and pruning (DHP), Partition algorithm, dynamic itemset counting (DIC) etc.[3] . FP-growth [4] is a well-known algorithm that uses the FP-tree data structure to get a condensed representation of the database transactions and employs a divide-and conquer approach to decompose the mining problem into a set of smaller problems. In spirit, it mines all the frequent itemsets by recursively determining all frequent 1-itemsets in the restrictive pattern base that is proficiently constructed with the help of a node link structure. In algorithm FP-growth-based, recursive production of the FP-tree affects the algorithm's complexity.

Most of the preceding work on association mining has utilized the conventional horizontal transactional database arrangement. However, a number of vertical mining algorithms have been proposed recently for association mining [5, 6, 9, 11, 12]. In a vertical database each item is associated with its equivalent tidset, the set of all transactions (or tids) where it appears. Mining algorithms using the vertical format have shown to be very valuable and usually do better than horizontal approaches. This advantage stems from the fact that frequent patterns can be counted via tidset intersections, instead of using complex internal data

structures (candidate generation and counting happens in a single step). The horizontal approach on the other hand needs complex search/hash trees. Tidsets offer ordinary pruning of extraneous transactions as a result of an intersection (tids not relevant drop out). Furthermore, for databases with lengthy transactions it has been shown using a simple cost model, that the vertical approach reduces the number of I/O operations [7]. In a current study on the integration of database and mining, the Vertical algorithm [8] was shown to be the best approach (better than horizontal) when forcefully integrating association mining with database systems. Eclat [9] is the primary algorithm to find frequent patterns by a depth-first search and it has been shown to execute fine. They use vertical database representation and count the support of itemset by using the intersection of tids. However, pruning used in the Apriori algorithm is not applicable during the candidate itemsets generation due to depth-first search. VIPER [5] uses the vertical database layout and the intersection to accomplish a excellent performance. The only difference is that they use the compacted bitmaps to represent the transaction list of each itemset. However, their compression method has limitations especially when tids are uniformly distributed. Zaki and Gouda [10] developed a new approach called dEclat using the vertical database representation. They store the difference of tids, called diffset, between a candidate k-itemset and its prefix k -1 frequent itemsets, instead of the tids intersection set, denoted here as tidset. They calculate the support by subtracting the cardinality of diffset from the support of its prefix k-1 frequent itemset. This algorithm has been exposed to gain significant performance improvements over Eclat. However, diffset will drop its advantage over tidset when the database is sparse.

Most of the preceding work on mining frequent patterns is based on the horizontal illustration. However, recently a number of vertical mining algorithms have been projected for mining frequent itemsets. Mining algorithms using the vertical representation have exposed to be effective and usually do better than horizontal approaches [11]. This advantage stems from the fact that frequent patterns can be counted via tidset intersections, instead of using complex internal data structures like the hash/search trees that the horizontal algorithms require [10]. The candidate generation and counting phases are done in a single step in vertical mining. This is done because vertical mining offers ordinary pruning of irrelevant transactions as a result of an intersection.

Another characteristic of vertical mining is the utilization of the autonomy of classes, where each frequent item is a class that contains a set of frequent k-itemsets (where k > 1) [6]. The vertical arrangement appears to be a natural choice for achieving association rule mining's objective of discovering correlated items. Computing the supports of itemsets is simpler and faster with the vertical arrangement since it involves only the intersections of tid-lists or tid-vectors, operations that are well-supported by existing database systems. In contrast, complex hash-tree data structures and functions are required to perform the same function for horizontal layouts. There is an automatic reduction of the database before each scan in that only those itemsets that are significant to the following scan of the mining process are accessed from disk. In the horizontal layout, however, irrelevant information that happens to be part of a row in which useful information is present is also transferred from disk to memory. This is because database reductions are comparatively hard to implement in the horizontal arrangement. Further, even if reduction were promising, the irrelevant information can be removed only in the scan following the one in which its irrelevance is discovered. Therefore, there is always a reduction delay of at least one scan in the horizontal layout.

Most of the preceding work on vertical mining concentrates on intersection of transaction [12]. This is based on intersection of perpendicular tid-vector where it is a set of columns with each column storing an IID and a bit-vector of 1's and 0' to represent the occurrence or nonexistence, respectively, of the item in the set of customer transactions. If we use list-based layout then it takes much less space than the bit-vector approach (which has the overhead of openly representing absence) in sparse databases. We make the case in this paper and use list-based layout [16]. To find intersection we use dynamic technique instead of traditional approach. We suggest a novel dynamic algorithm for frequent pattern mining in which we generate transaction pair and for generating frequent pattern we find out by longest common subsequence using dynamic function

3. Dynamic Function

The longest common subsequence problem is one of the frequent problems which can be solved powerfully using dynamic programming. "The Longest common subsequence problem is, we are given two sequences X=<x1,x2----------xn> and Y=<y1,y2---------ym> and wish to find a maximum length

common subsequence of X and Y" for example : if X=<A,B,C,B,D,A,B> and Y=<B,D,C,A,B,A> then The sequence <B, C, B, A> longest common subsequence. Let us define CC [i, j] to be the length of an LCS of the sequences xi and yj. If either i=0 or j=0, one of the sequence has length 0, so the LCS has length 0. The Optimal substructure of the LCS Problem gives the recursive formula in fig.1

$$C(i,j)=\begin{cases}0 & if\ i=0\ or\ j=0\\ C(i-1,j-1)+1 & if\ i,j>0\ and\ xi=yj\\ \max\left(c(i,j-1),c(i-1,j)\right) & if\ i,j>0\ and\ xi\neq yj\end{cases}$$

Figure 1. Longest Common Subsequence Recursive Formula

4. Algorithm

The Novel algorithm works over the entire database file, now apply Apriori like Algorithm in which first we generate transaction pair with longest common subsequence of item id instead of item id pair. For each Iteration we apply following sequence of operation until condition occurred. First generate the transaction pair and prune with empty longest common subsequence by dynamic function. To count the support , instead of whole database for each pruned pattern we find all subset and display it and also stored new transaction pair and its attribute common subsequence so that next iteration we trace above subsequence. To find longest common subsequence we used dynamic function which faster then traditional function. Write pruned transaction pair list with attribute common subsequence so that in next pass we used this pair list instead of all pair list. An advantage of This approach is in each iteration database filtering and reduces, so each iteration is faster then previous iteration

Algorithm FPMDF (Frequent Patterns Mining Using Dynamic Function)

I. Given Database T with ∂(Min. no. of transaction)
II. K: =2.
III. While Lk-1≠ { } do
IV. Ck=Compute each pair of each previous transaction pair .
V. Computer LCS of Item id for each previous transaction pair.
VI. Lk=Prune Transaction Pair having empty LCS.
VII. If ∂<=k then Fk=All_Subset(Lk)
VIII. K:=K+1

5. Explanation with example which support the arguments

Study the following transaction database .T={T1,T2,T3,T4,T5 >, Assume σ=40%, Since T contains 5 records, it means that an itemset that is supported by at least two transactions is a frequent set and output shown in fig.2

TABLE I. GIVEN DATASE T (C1)

TId	Attributes (Item Id)
1	1,2,5,6,7,9,10,15
2	1,3,14
3	2,3,5,6,7,8,9,12
4	4,10,15
5	2,4,5,7,9,11,13

Now Apply Algorithm

Iteration 1

Generate Transaction Pair with two elements with Longest Common Subsequence (LCS) By Dynamic Function of Attributes

TABLE II. C2

TId	Attributes (Item Id)
1 ,2	1
1,3	2,5,6,7,9
1,4	10,15
1,5	2,5,7,9
2,3	3
2,4	NIL
2,5	NIL
3,4	NIL
3,5	2,5,7,9
4,5	4

Prune C2 by removing Transaction pair having Empty LCS of attributes.

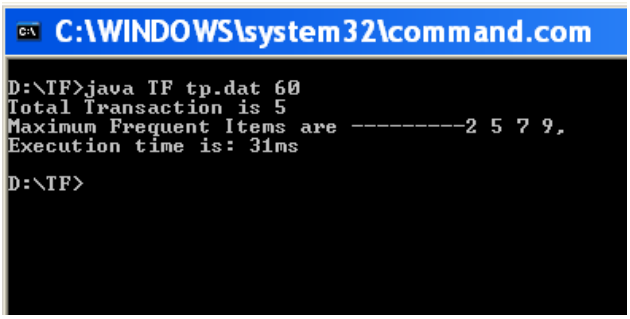

Figure 2. Frequent Pattern with support 60%

TABLE III. L2

TId	Attributes (Item Id)
1 ,2	1
1,3	2,5,6,7,9
1,4	10,15
1,5	2,5,7,9
2,3	3
3,5	2,5,7,9
4,5	4

If σ=40%, frequenct set support record=2 then

F2=All_Subset(L2)

F2 :={ 1,2, 3, 4, 5, 6, 7,9,10,15,(2,5),(2,6),(2,7),(2,9),(5,6),(5,7),(5,9),(6,7),(6,9),(7,9),(10,15),(2,5,6),(,2,5,7),(2,5,9),(2,6,7),(2,6,9),(2,7,9),(2,5,6,7),(2,5,6,9),(2,5,7,9), (2,6,7,9),(5,6,7,9),(2,5,6,7,9)}.

Iteration 2

Generate Transaction Pair with 3 elements with Longest Common Subsequence (LCS) By Dynamic Function of Attributes

TABLE IV. C3

TId	Attributes (Item Id)
1 ,2,3	NIL
1,2,4	NIL
1,2,5	NIL
1,3,4	NIL
1,3,5	2,5,7,9
1,4,5	NIL

Prune C2 by removing Transaction pair having Empty LCS of attributes.

TABLE V. L3

TId	Attributes (Item Id)
1,3, 5	2,5,7,9

If σ=40%, frequenct set support record=2 then

F3=F2 U All_Subset(L3)

F3:={1,2,3,4,5,6, 7,9,10,15,(2,5),(2,6),(2,7),(2,9),(5,6),(5,7),(5,9),(6,7),(6,9),(7,9),(10,15),(2,5,6),(,2,5,7),(2,5,9),(2,6,7),(2,6,9),(2,7,9),(2,5,6,7),(2,5,6,9),(2,5,7,9),(2,6,7,9),(5,6,7,9),(2,5,6,7,9) }

If σ=60%, frequenct set support record=3 then

F3= All_Subset(L3)

F3 :={ 2,5, 7,9,(2,5), ,(2,7),(2,9), (5,7),(5,9),(7,9), ,(2,5,7),(2,5,9), ,(2,7,9),(5,7,9),(2,5,7,9), }

6. Experimental results

In this section we performed a set of experiments to evaluate the effectiveness of the frequent pattern mining using dynamic function method. The algorithm DFPMT was executed on a Pentium 4 CPU, 2.26GHz, and 1 GB of RAM computer. It was implemented in Java. The experiment database sources are T40I4D100K, provided by the QUEST generator of data generated from IBM's Almaden lab. The experimental dataset contains data whose records are set to 10. The testing results of experiments are showed in Fig.3. In the Fig.3, the horizontal axis represents the number of support in database and the vertical axis represents mining time. The three curves denote different time cost of the algorithm Apriori, FP Growth and FPMDF with different minsup.

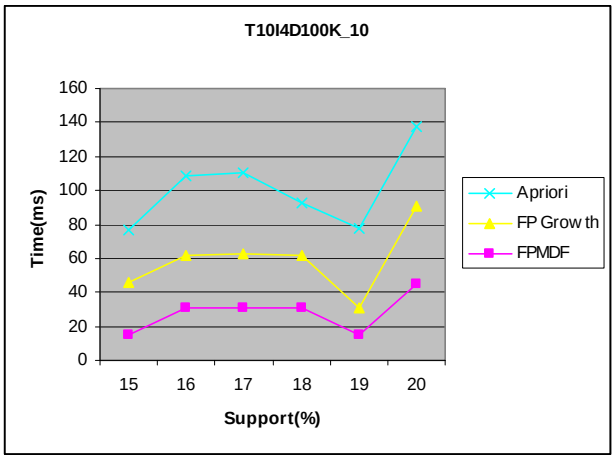

Figure 3. The test results of apriori, FP Growth and DFPMT

6. Conclusion

Discovering frequent objects (item sets, sequential patterns) is one of the most vital fields in data mining. It is well understood that it require running time and memory for defining candidates and this is the motivation for developing large number of algorithm.. We presented a new research trend on frequent pattern mining in which if the number of transaction are very less as compare to attributes or items specially in medical fields then instead of generating item id pair we generate pair of transaction with longest common subsequence of item ids. Then we gave an approach to use this framework to mine all the itemset satisfying. We used dynamic function which is superior to conventional function for finding longest common subsequence. We also presented a new research trend on filtering the database in all iteration. Further investigations are required to clear the possibilities of this method

Acknowledgments

We thank Sh. R. S. Thakur and Sh. K K Shrivastava for discussing and giving us advice on its implementation.

References

[1] R. Agrawal, T. Imielinski, and A.N. Swami, "Mining Association Rules between Sets of Items in Large Databases," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 207-216, May 1993.

[2] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," Proc. 20th Int'l Conf. Very Large Data Bases, pp. 487-499, 1994.

[3] B. Goethals, "Survey on Frequent Pattern Mining," manuscript, 2003.

[4] 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.

[5] P. Shenoy, J.R. Haritsa, S. Sudarshan, G. Bhalotia, M. Bawa, and D. Shah. Turbo-charging vertical mining of large databases. In *ACM SIGMOD Intl. Conf. Management of Data*, May 2000.

[6] M. J. Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):372-390, May-June 2000.

[7] B. Dunkel and N. Soparkar. Data organization and access for efficient data mining. In *15th IEEE Intl. Conf. on Data Engineering*, March 1999.

[8] S. Sarawagi, S. Thomas, and R. Agrawal. Integrating association rule mining with databases: alternatives and implications. In *ACM SIGMOD Intl. Conf. Management of Data*, June 1998.

[9] M.J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, "New Algorithms for Fast Discovery of Association Rules," Proc. Third Int'l Conf. Knowledge Discovery and Data Mining, pp. 283-286, 1997.

[10] M.J. Zaki and K. Gouda, "Fast Vertical Mining Using Diffsets," Proc. Ninth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining, pp. 326-335, 2003.

[11] M. Song, S. Rajasekaran. (2006). "A Transaction Mapping Algorithm for Frequent Itemsets Mining" , IEEE Transactions on Knowledge and Data Engineering , Vol.18, No.4, pp. 472-481, April 2006.

[12] M. jamali, F. taghiyareh (2005) "Generating Frequent Pattern through Intersection between Transactions

[13] Sunil Joshi , Dr. R. C. Jain : accepted and published research paper in The IEEE 2010 International Conference on Communication software and Networks (ICCSN 2010) *on "A Dynamic Approach for Frequent Pattern Mining Using Transposition of Database" from* 26 - 28 February 2010

[14] Finding Longest Increasing and Common Subsequences in Streaming Data David Liben-Nowell_ y dln@theory.lcs.mit.edu Erik Vee_ z env@cs.washington.edu An Zhu_ x anzhu@cs.stanford.edu November 26, 2003

[15] B. Jeudy and F. Rioult, Database transposition for constrained closed pattern mining, in: Proceedings of Third International Workshop on Knowledge Discovery in Inductive Databases (KDID) co-located with ECML/PKDD, 2004.

[16] M. J. Zaki and C. J. Hsiao. CHARM: An e±cient algorithm for closed itemset mining. In Proc. 2002 SIAM Int. Conf. Data Mining (SDM'02), pages 457{473, Arlington, VA, April 2002.

Sunil Joshi is presently working as a Ass. Professor, Computer Applications at Samrat Ashok Technological Institute Vidisha (M.P). He has 9 years teaching experience and 2 years research experience. His research areas include Data mining.

R S Jadon is presently working as a Head, Computer Applications at Madhav Institute of Technology and Science, Gwalior. He has 12 years research experience. He has presented research papers in more than 30 national and international conferences and published more than 30 papers in national and international journals. His research areas include Video Data Processing.

R C Jain is presently working as a Director and Head; Computer Applications at Samrat Ashok Technological Institute Vidisha He has 30 years teaching experience and 15 years research experience. He has presented research papers in more than 100 national and international Conferences and published more than

150 papers in national and international journals. His research areas include Data mining and Network security.