International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 1 Issue 5, July - 2012

An Efficient Algorithms for Generating Frequent Pattern Using Logical Table With AND, OR Operation

Kamlesh Malpani
Ph.D. Scholar,
Department of Computer Engineering,
Pacific Academy of Higher Education and Research, Udaipur (Raj.)

Dr.Parashu.Ram Pal
PG Department of Computer Application,
Ajay Kumar Garg Engineering College
Ghaziabad, U.P. India

Abstract:- Frequent Pattern Mining plays an essential role in many data mining tasks that try to find interesting patterns from databases, such as association rules, correlations, Market basket analysis is a useful method of discovering customer purchasing patterns by extracting associations or co-occurrences in transactional databases[1,2]. Information obtained from the analysis can be used in marketing, sales, service, and operational strategies, In this paper, we propose a new algorithm based Logical Operation (AND,OR). In this algorithms we are using simple Logical operation (AND, OR) on data set containing items. We use simple table to perform AND, OR operation to avoid joining and pruning. The advantage of this new technique is fast operation on dataset containing items and provides facilities to avoid unnecessary scans to the database

Keywords: Logical Operation AND, OR, Corelation

I. INTRODUCTION

Data mining is the process of extracting patterns from data. It is becoming as an increasingly important tool to transform these data into information. Frequent item sets mining is a popular and important, first step in data mining for analyzing data sets across a broad range of applications. It plays an essential role in many important data mining tasks.[1,2,3] Let I = { I1, I2, I3, …, Im} be a set of items. Let D be the transactional database where each transaction T is a set of items. Each transaction is associated with an identifier TID [3]. A set of items is referred as item set. An item set that contains K items is a K-item set. The number of transactions in which a particular item set exists gives the support or frequency count or count of the item set. If the support of an item set I satisfies the minimum support threshold, then the item set I is a frequent item set. classified based on the completeness of patterns to be mined, the levels of abstraction involved in the rule set, the number of data dimensions involved in the rule, the types of values handled in the rule, the kinds of rules to be mined, the kinds of patterns to be mined[2,4,5]. The classification of algorithms for frequent **item set** mining is Apriori-like algorithms, frequent pattern growth based algorithms it is impractical to generate the entire set of frequent item sets for the very large databases. There is much research on methods for generating all frequent item sets efficiently [3]. Most of these algorithms use a breadth-first approach, i.e. finding all k-item sets before considering (k+1) item sets. The performance of all these algorithms gradually degrades with dense datasets [2,3]

The main drawback of frequent item sets is they are very large in number to compute or store in computer. This leads to the introductions of closed frequent item sets and maximal frequent item sets. An item set X is closed in a data set S if there exists no proper superitemset Y such that Y has the same support count as X in S. An item set X is closed frequent item set in set S if X is closed and frequent in S. an item set X is a maximal frequent item set in set S if X is frequent and there exists no super-item set Y such that X . Y and Y is frequent in S. Maximal frequent item set mining is efficient in terms of time and space when compared to frequent item sets and closed frequent item sets because both are subsets of maximal frequent item set. Some of the algorithms developed for mining maximal frequent [5]

II APRIORI ALGORITHMS

Apriori algorithm is an influential algorithm for mining frequent item sets for Boolean association rules. It uses a Level-wise search, where k-item sets

(Anitemset that contains k items is a k-item set) are used to explore (k+1)-item sets, to mine frequent item sets from transactional database for Boolean association rules[4].

First, the set of frequent 1-itemsets is found. This set is denoted L1. L1 is used to find L2, the frequent 2-itemsets, which is used to find L3, and so on, until no more frequent k-item sets can be found. The finding of each Lk requires one full scan of the database.

Apriori property: All non-empty subsets of a frequent item set must also be frequent. [1,6,5]

It performs the following tasks:

1. Reducing the search space to avoid finding of each Lk requires one full scan of the database
2. If an item set I does not satisfy the minimum support threshold, min_sup, the I is not frequent, that is, P (I) < min_sup
3. If an item A is added to the item set I, then the resulting item set (i.e., IUA) cannot occur more frequently than I. Therefore, I UA is not frequent either, that is, P (I UA) < min_sup.

A two step process is followed, consisting of join and prune actions [5].

1. The join step: To find Lk, a set of candidate k-item sets is generated by joining Lk-1 with itself. This set of candidates is denoted Ck. The join, Lk-1 with Lk_1, is performed, where members of Lk-1 are joinable if they have (k_2) items in common.
2. The prune step: Ck is a superset of Lk, that is, its members may or may not be frequent, but all of the frequent k-item sets are included in Ck. [4,5]A scan of the database to determine the count of each candidate in Ck would result in the determination of Lk (i.e., all candidates having a count no less than the minimum support count are frequent by definition, and therefore belong to Lk). Ck, however, can be huge, and so this could involve heavy computation. To reduce the size of Ck, the Apriori property is used as follows. Any (k-1)-item set that is not frequent cannot be a subset of a frequent k-item set. Hence, if any (k-1)-subset of a candidate k-item set is not in Lk_1, then the candidate cannot be frequent either and so can be removed from Ck. This subset testing can be done quickly by maintaining a hash tree of all frequent items sets [5, 14].

A. Limitations

1. The algorithm is of low efficiency, such as firstly it needs to repeatedly scan the database, which spends much in I/O.
2. Secondly, it creates a large number of 2- candidate item sets during outputting frequent 2- item sets.
3. Thirdly, it doesn’t cancel the useless item sets during outputting frequent k- item sets.[2,5,7]

B. Methods to Improve Apriori’s Efficiency[5,6]

•*Hash-based item set counting*: A k-item set whose corresponding hashing bucket count is below the threshold cannot be frequent.

•*Transaction reduction*: A transaction that does not contain any frequent k-itemsetis useless in subsequent scans.[

•*Partitioning:* Any item set that is potentially frequent in DB must be frequent in at least one of the partitions of DB.

•*Sampling*: mining on a subset of given data, lower support threshold + a method to determine the completeness.

•*Dynamic item set counting*: add new candidate item sets only when all of their subsets are estimated to be frequent.

III. PROPOSED ALGORITHM

Our algorithm is an effective algorithm for mining association rules in large databases .Like the Apriori algorithm, our algorithm mines association rules in two steps. In the first step compute frequent item sets using logic OR and AND operations. The Implemented algorithm gains significant performance improvement over the Apriori algorithm.[8,9]

A. Generation of Frequent Item sets

The implemented algorithm generates frequent itemsets through evolutionary iterations based on two tables, the item details table and the transaction table.

B. Transforming a transaction details into a logical Table

The Logical table with element values of 1 or 0 ,where items are present in the transaction means 1 otherwise 0. Finally, a column vector Ck is utilized to store the reference count of all frequent k-item sets in the kth iteration. The reference count on a k-item set can be obtained by counting the number of 1‟ s in the corresponding row of logical table.[10]

C. Generation of Frequent k-item sets:

Frequent k-item sets can be generated through the following iteration:

Repeat

1. Read a pair of different rows from a logical table.
2. go to step 3 (i.e., until a new k-item set has been found).
3. Performing AND operation on the two rows of Logical table, correspond to the rows of step2. The result shows that, which transactions contain this new k-item set. And then counting the number of 1‟s in the result to get the reference count of this new k-item set. If the count is less than the number of transactions required by the minimum support, the new k-item set is discarded. After the generation of frequent k-item set, the logical table of the k-item set and its corresponding reference count vector are kept in frequent item set table for generating association rules.[12,13]

IV IILUSTRATIVE EXAMPLE

A. Ariori Algorithms

Table containing transactions

TABLE 1

TID	List of Items
T100	I1,I2,I5
T200	I2,I4
T300	I2,I3
T400	I1,I2,I4
T500	I1,I3
T600	I2,I3
T700	I1,I3
T800	I1,I2,I3,I5
T900	I1, I2,I3

Table with 1 item set and support count

TABLE 2

Item Set	Sup_Count
I1,I2	4
I1,I3	4
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2

Table with 1 item set and minimum support count

TABLE 3

Item Set	Sup_ Count
I1	6
I2	7
I3	6
I4	2
I5	2

Table with 1 item set and minimum support count

TABLE 4

Item Set
I1,I2
I1,I3
I1,I4
I1,I5
I2,I3
I2,I4
I2,I5
I3,I4
I3,I5
I4,I5

Table with 2 item set

TABLE 5

Item Set	Sup_Count
I1,I2	4
I1,I3	4
I1,I4	1
I1,I5	2
I2,I3	4
I2,I4	2
I2,I5	2
I3,I4	0
I3,I5	1
I4,I5	0

Table with 2item set and support count

TABLE 6	
Item Set	**Sup_Count**
I1,I2	**4**
I1,I3	**4**
I1,I5	**2**
I2,I3	**4**
I2,I4	**2**
I2,I5	**2**

Table with 2 itemset and minimum support count

TABLE 7
Item set
I1,I2,I3
I1,I2,I5

with frequent item set

B Proposed Algorithms

Table containing transactions

TABLE 8	
TID	**List of Items**
T100	**I1,I2,I5**
T200	**I2,I4**
T300	**I2,I3**
T400	**I1,I2,I4**
T500	**I1,I3**
T600	**I2,I3**
T700	**I1,I3**
T800	**I1,I2,I3,I5**
T900	**I1, I2,I3**

TABLE 9					
	Items				
	I1	**I2**	**I3**	**I4**	**I5**
I1	**1**	**0**	**0**	**0**	**0**
I2	**0**	**1**	**0**	**0**	**0**
I3	**0**	**0**	**1**	**0**	**0**
I4	**0**	**0**	**0**	**1**	**0**
I5	**0**	**0**	**0**	**0**	**1**
T100	**1**	**1**	**0**	**0**	**1**
T200	**0**	**1**	**0**	**1**	**0**
T300	**0**	**1**	**1**	**0**	**0**
T400	**1**	**1**	**0**	**1**	**0**
T500	**1**	**0**	**1**	**0**	**0**
T600	**0**	**1**	**1**	**0**	**0**
T700	**1**	**0**	**1**	**0**	**0**
T800	**1**	**1**	**1**	**0**	**1**
T900	**1**	**1**	**1**	**0**	**0**
Support Count	**6**	**7**	**6**	**2**	**2**

Table with item present or absent in transaction
Also with their support count

TABLE 10						
Two Item set	**I1, I2**	**I1, I3**	**I1, I5**	**I2, I3**	**I2, I4**	**I2, I5**
I1	**1**	**1**	**1**	**0**	**0**	**0**
I2	**1**	**0**	**0**	**1**	**1**	**1**
I3	**0**	**1**	**0**	**1**	**0**	**0**
I4	**0**	**0**	**0**	**0**	**1**	**0**
I5	**0**	**0**	**1**	**0**	**0**	**1**
T100	**1**	**0**	**1**	**0**	**0**	**1**
T200	**0**	**0**	**0**	**0**	**1**	**0**
T300	**0**	**0**	**0**	**1**	**0**	**0**
T400	**1**	**0**	**0**	**0**	**1**	**0**
T500	**0**	**1**	**0**	**0**	**0**	**0**
T600	**0**	**0**	**0**	**1**	**0**	**0**
T700	**0**	**1**	**0**	**0**	**0**	**0**
T800	**1**	**1**	**1**	**1**	**0**	**1**
T900	**1**	**1**	**0**	**1**	**0**	**0**
Sup count	**4**	**4**	**2**	**4**	**2**	**2**

Table with 2 Itemset with support count

TABLE 11		
Three item set	**I1,I2,I3**	**I1,I2,I5**
I1	**1**	**1**
I2	**1**	**1**
I3	**1**	**0**
I4	**0**	**0**
I5	**0**	**1**
T100	**0**	**1**
T200	**0**	**0**
T300	**0**	**0**
T400	**0**	**0**
T500	**0**	**0**
T600	**0**	**0**
T700	**0**	**0**
T800	**1**	**1**
T900	**1**	**0**
Sup Count	**2**	**2**

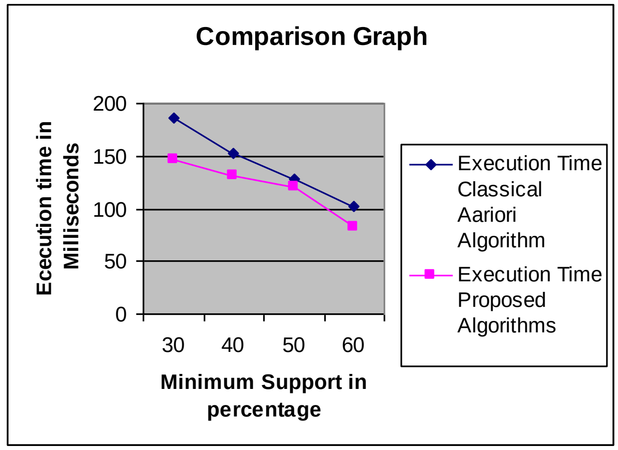
Table with frequent item set

V COMPARISON WITH GRAPH

In order to show the performance of the proposed algorithm, we conducted an experiment using the Apriori algorithm and proposed algorithm. The algorithms were implemented in Dot Net and tested on a Windows XP platform. The test database are taken from easy day shopping mall The number of items N is set to 20; D is the number of transactions which 100; T is the averages size of transactions, and I is the average size of the maximum frequent item sets. Graph show results for different numbers of minimum supports. The results show that the performance of our algorithm is much better than that of the Apriori algorithm. This is because the greater the minimum support, the more less candidate item sets the Apriori algorithm has to determine, and also the Apriori algorithm join and pruning processes take more time to execute. However, and it spends less time calculating k-supports with the logical item table.

VI CONCLUSION

The most common application of association rule mining is market basket analysis. In this paper, An Efficient algorithm for mining association rules using Logical Table based approach is proposed. The main features of this algorithm are that it only scans



the transaction database once, it does not produce candidate item sets, and In addition, it stores all transaction data in bits, so it needs less memory space and can be applied to mining large databases

VII REFERENCES

[1] Krishnamurthy M, "Frequent item set generation using hashing-quadratic probing technique" European Journal of Scientific Researchissn 1450-216x ,Vol.50 no.4 ,2011, pp. 523-532.

[2] Arora T., Yadav R., "Improved Association Mining Algorithm for large Dataset International Journals of Computational Engineering & Management, vol. 13, july 2011 issn (online): 2230-7893 www.ijcem.org

[3] Singh Vaibhav Kant, ShahVijay,Jain Yogendra Kumar Shukla Anupam, Thoke A.S.Singh Vinay Kumar,Dule Chhaya,Parganiha Vivek "Proposing an efficient method for frequent pattern mining ".

[4] Dai Jja-yu,Yang Don-lin,Wu jungpin and Hung Ming-chuan "An efficient data mining approach on compressed transactions" in proceedings of world academy of science, engineering and technology volume 30 july 2008 issn 1307-688433

[5] Wu Huan, Lu Zhigang, Pan Lin, Xu Rongsheng Xu,Jjang Wenbaq" An improved apriori-based algorithm for association rules mining" 2009 sixth international conference on fuzzy systems and knowledge discovery

[6] Goswami D.N. Chaturvedi Anshu. Raghuvanshi C.S.,"An algorithm for frequent pattern mining based on apriori" (ijcse) International Journal on Computer Science and Engineering vol. 02, no. 04, 2010, 942-947

[7] Prof.Dr.Patnaik Prashant Mr.Padhi Sanjay, "An Efficient Algorithm for Mining of Frequent Items using Incremental Model"

[8] Amornchewin Ratchadaporn "Mining Dynamic Databases using probability-based Incremental Association Rule Discovery Algorithm" by journal of universal computer science, , submitted: 15/12/08, accepted: 25/6/09, appeared: 28/6/09 j.ucs, vol. 15, no. 12 (2009) 2409-2428

[9] Taha Ahmed, Taha Mohamed, Nassar Hamed,Gharib Tarek F "Darm: decremental Association Rules Mining" journal of intelligent learning systems and applications, 2011, 3, 181-189 doi:10.4236/jilsa.2011.33019 published online august 2011 (http://www.scirp.org/journal/jilsa)

[10] Karam Gouda "Genmax: An Efficient Algorithm for Mining Maximal Frequent Item sets Data Mining and Knowledge Discovery", 11, 1–20, 2005 _c 2005 springer science + business media, inc. manufactured in the netherlands. department of mathematics, faculty of science, benha, egypt mohammed j. zak

[11] Antonie Maria-Luiza, Za¨Tane Osmar R "Mining Positive and Negative Association Rules: An Approach for Confined Rules" department of computing science, university of alberta

[12] Dong Wuzhou, Ren Jjadong, Gaitaq Juan Yi "An Incremental Algorithm for Frequent Itern Mining Based on Bit-Sequence ".

[13] Sujatha D., Deekshatulu B.L., "Algorithm for Mining Time Varying Frequent Itemsets " Journal of Theoretical and Applied Information Technology © 2005 - 2009 jatit. all rights reserved.

[14] Amornchewin Ratchadaporn "Mining Dynamic Databases using Probability-based Incremental Association Rule Discovery Algorithm" Journal of Universal Computer Science, vol. 15, no. 12 (2009), 2409-2428 submitted: 15/12/08, accepted: 25/6/09, appeared: 28/6/09 j.ucs