National Conference on Emerging Trends in Computer, Electrical & Electronics (ETCEE-2015)
International Journal of Advance Engineering and Research Development (IJAERD)
e-ISSN: 2348 - 4470 , print-ISSN:2348-6406,Impact Factor:3.134

Survey Conducted On Algorithms For Finding Frequent And Closed Frequent Itemset With Incremental Approach

Nidhi Dangar[1], Trupti Kodinariya[2]

[1,2]Computer Department, Gujarat Technological University
Address
[1]nidhidangar@gmail.com
[2]trupti.kodinariya@gmail.com

***Abstract*— Data Mining (also known as Knowledge Discovery from Database KDD) is defined as extracting required knowledge from large available data. Association rule mining is defined as finding correlation among various items in available large dataset and finding useful knowledge and patterns from them. Frequent itemset is defined as finding the items with more occurrences in the dataset than other items. In recent time , data mining is an emerging field as execution speed and time consumption with incremental database is highly demanded .In this paper , a survey is conducted over association rule mining and frequent and closed frequent itemset , that how much work done in these fields recently and before. The basic purpose behind the survey is to compare different approaches and find one better approach which can efficiently find set of frequent and closed frequent item with incremental database.**

***Keywords*— Association rule mining , Closed Frequent Itemset Data Mining ,Frequent Itemset, Incremental database.**

I. INTRODUCTION

Data mining or knowledge discovery in databases (KDD) is a collection of explorations techniques based on advanced analytical methods and tools for handling a large amount of information [4].During recent years, one of active research topic is association rule discovery. The association rule discovery is used to identify relationships among items in very large databases, to extract interesting correlations, associations among sets of items in the transaction databases or other data repositories. For example, given a market basket database, it would be interesting for decision support to know the fact that 30% of customers who bought cocoa powder and sugar also bought butter. This analysis may be used to provide some basis if is required to increase the sales and introduce from free schemes like , if 3 kg of sugar is bought then 100g butter free[2] .

There are two sub problems on association rule mining: Finding frequent or large itemsets is the first sub problem. Frequent items can be generated in two steps. Firstly, candidate large itemsets are generated and secondly frequent itemsets are generated using these candidate itemset. The itemset whose support is greater than the minimum support are referred as frequent itemsets [2].The support count for an itemset X is the total number of the transaction containing X. The support for an itemset X is defined as the ratio of support count to the total number of transaction in the database [13].

Another major recent concept in this area is finding closed frequent itemset. A frequent itemset is called closed if there does not exist a superset that has the same support. Closed frequent itemset even preserve knowledge of the support values. The reason is that each frequent itemset has uniquely determined closed superset with the same support [3].

Apriori algorithm is classical algorithm in data mining towards finding frequent itemset and discovering association rule. Modifications in these basic algorithms also provide efficient techniques for finding frequent itemset.

Next section will provide various approaches and/or algorithms in this wide area of data mining.

II. RELATED BACKGROUND

Apriori is a seminal algorithm for finding frequent itemsets using candidate generation [1].It is characterised as level-wise complete search algorithm using anti-monotonicity of itemsets, ”if an itemset is not frequent, any of its superset is never frequent”. By convention, Apriori assumes that items within a transaction or itemset are sorted in lexicographic order. Apriori first scans the database and searches for frequent itemsets of size 1 by accumulating the count for each item and collecting those that satisfy the minimum support requirement [19]. Apriori works in two steps i.e. join step and prune step. In join step, the itemset with higher support greater or equal to minimum support threshold will be moved to data base. In prune step, the itemset which cannot satisfy the minimum support threshold condition will be discarded.

The limitation with apriori algorithm is that it every time scans the database to find and store frequent itemset. It is highly time consuming. This approach may not work better with large database or with incremental database. There should be some modifications in the algorithm to work better with it.

In [6], Jaishree Singh et.al proposed an algorithm called Improved Apriori algorithm which reduces the scanning time

by cutting down unnecessary transaction records as well as reduce the redundant generation of sub-items during pruning the candidate itemset, which can form directly the set of frequent itemsets and eliminate candidate having a subset that is not frequent.

In this method, an attribute Size Of Transaction (SOT) is introduced. It contains number of items in individual transaction in database. The deletion process of transaction in database will made according to the value of K. Whatever the value of K, algorithm searches the same value for SOT in database. If value of K matches with value of SOT then delete only those transaction from database [6].

This improved algorithm works better than apriori and also reduces I/O spending by reducing transaction records from database. Limitation of it cannot work well with incremental database.

In [10], Marghany H. Mohammed and Mohammed M, Darwieesh proposed two algorithms CTFI(Count Table FI) and BCTFI (Binary Count Table FI) for efficiently finding frequent itemset.

In the CTFI algorithm, a countable is used to compress database for quick frequent itemset generation and support count, respectively. It transforms original transaction data into new smaller transaction data with all information of frequent itemsets. By this pass over the database with given transactions, we can get a new merge transactions and then the frequent itemsets can be obtained from building countable of all items in new merge transactions, CTFI uses Intersection operation to generate frequent itemsets based on countable which compress the items [10].

Table 1 Transaction Data D in ARM

TID	List of Items
T1	A,B,E
T2	B,D
T3	B,C
T4	A,B,D
T5	A,C
T6	B,C
T7	A,C
T8	A,B,C,E
T9	A,B,C

Table 2 New Transaction data

TID	List of Items
T1	A,B,E
T2	B,D
NT3	B,C
T4	A,B,D
NT5	A,C
NT8	A,B,C,E

Table 3 Count Table of all items in D

TID	A	B	C	D	E
T1	1	1	0	0	1
T2	0	1	0	1	0
NT3	0	2	2	0	0
T4	1	1	0	1	0
NT5	2	0	2	0	0
NT8	2	2	2	0	1

Table 4 Frequent itemset in D

Itemset	Supp.count
A	6
B	7
C	6
D	2
E	2
A,B	4
A,C	4
A,E	2
B,C	4
B,D	2
B,E	2
A,B,C	2
A,B,E	2

In BCTFI algorithm, the original transaction data is represented by 0/1 binary representation. Using this representation, we are able to transform the data to decimal number [10].

If the set of items of each transaction can be transformed in simple counTable, it may be possible to avoid repeatedly

scanning the original transaction database. If multiple transactions share a common prefix 0's and 1's according to some sorted order in binary code, it may be possible to merge the shared sets with the number of occurrences registered as count with items listed according to a fixed order. It is easy to check whether two transactions are identical if they have the same decimal number representation [10].

Table 5 Binary representation of data

TID	A	B	C	D	E	Decimal
T1	1	1	0	0	1	25
T2	0	1	0	1	0	10
T3	0	1	1	0	0	12
T4	1	1	0	1	0	26
T5	1	0	1	0	0	20
T6	0	1	1	0	0	12
T7	1	0	1	0	0	20
T8	1	1	1	0	1	29
T9	1	1	1	0	0	28

Table 6 New Transaction in D

TID	A	B	C	D	E	Decimal
T1	1	1	0	0	1	25
T2	0	1	0	1	0	10
NT3	0	1	1	0	0	12
T4	1	1	0	1	0	26
NT5	1	0	1	0	0	20
NT8	1	1	1	0	1	29

Table 7 Content Table of all items in D

TID	A	B	C	D	E
T1	1	1	0	0	1
T2	0	1	0	0	1
NT3	0	2	2	0	0
T4	1	1	0	1	0
NT5	2	0	2	0	0
NT8	2	2	2	0	1

Limitation with these two algorithms is that it cannot maintain the items over incremental database, but both can efficiently find frequent itemset.

Moving towards closed frequent itemset mining, In [12], R Uday Kiran and P Krishna Reddy proposed an approach called IMS(Improved Multiple Support) apriori approach which is an extended approach of MS apriori approach.In this approach, each item is assigned with a minsup value known as "Minimum Item Support"(MIS) and frequent itemsets are generated if an itemset satisfies the lowest MIS value among the respective items. The MIS value is assigned to each item equal to a percentage of its support. For every item $i_j \in I$, the $MIS(i_j)$ is calculated as:

$MIS(i_j) = \beta S(i_j)$, if $\beta S(i_j) > LS$

$= LS$ else (1)

Where β is user specified proportional value which can be varied between 0 to 1.$S(i_j)$ refers to support of an each item equal to $f(i_j)/N$ ($f(i_j)$ is frequency if i_j and N is number of transactions in transaction dataset).LS is user specified Least Support Value[12].

Problem with this approach is if β is set high, it can be observed that MIS for rare items will be relatively more close(almost equivalent) to their supports as compared with frequent items.As a result, itemsets containing rare items fail to satisfy the support of rare item in that itemset. So, frequent itemsets involving rare items are missed [12].

IMS approach use the notion of support difference SD to specify the minimum support to items Support Difference refers acceptable deviation of an item from its frequency so that an itemset involving that item can be considered as frequent itemset. For each item i_j ,

$MIS(i_j) = S(i_j) - SD$ when $(S(i_j)-SD) > LS$

$= LS$ otherwise (2)

Value of SD can be calculated as $SD=\lambda(1-\alpha)$ (3)

λ = parameter like mean, median, mode, max support of an item

α = parameters ranging between 0 to 1. [12].

IMS apriori approach efficiently prunes the itemsets in such a way of no losing rare itemset. But due to applying support for each item, it may be difficult to handle incremental approach.

In [13], Show-Jane Yen et.al. presented an algorithm called MRFI-del (Maintenance of Representative Frequent Itemset). This algorithm works for maintaining closed frequent itemsets when transactions are deleted from a transaction database. When a transaction is deleted from a transaction database, MRFI-del only needs to apply some rules to determine which closed itemsets need to be updated without generating the subsets of the transactions and searching the supersets for each subset.

There are two structures for MRFI-del algorithm: First is content table and second is item table. Content table records the information about closed itemsets.

Table 8 Transaction Database D

TID	Items Bought
1	ACTW
2	CDW
3	ACTW
4	ACDW
5	ACDTW

6	CDT

Table 9 Content Table

ID	Closed Itemset	SC	IDs of I.C. Subsets	IDs of I.C. Superset
1	ACTW	3	5,9	6
2	CDW	3	3,8	4
3	CW	5	10	2,5
4	ACDW	2	2,5	6
5	ACW	4	3	1,4
6	ACDTW	1	1,4,7	NULL
7	CDT	2	8,9	6
8	CD	4	10	2,7
9	CT	4	10	1,7
10	C	6	NULL	3,8,9

The second structure is item table which includes two fields Item and identifier of the closed itemsets (CIDs) that contain the item. When a new closed itemset is generated, MRFI assigns it a unique identifier and records the identifier in the field CIDs of the item table for the items which are contained in the new closed itemsets.

Table 10 Item Table

Item	CIDs
A	1,4,5,6
C	1,2,3,4,5,6,7,8,9,10
T	1,6,7,9
W	1,2,3,4,5,6
D	2,4,6,7,8

MRFI-del algorithm is updated when transaction T is deleted. Because only the support for each closed itemset X C T will be deceased MRFI-del only needs to check if X is still a closed itemset after deleting T. If X is still a closed itemset, then we only need to subtract the original support count of X by 1.If X turns out to be non-closed itemset. We have to delete them from content table and item table [13].

MRFI-del avoids doing a so many searching calculation and out performs CFI-del (Closed Frequent Itemset) algorithm significantly. The only limitation is there is no specification about incremental database handling.

In [11], Ratchadaporn Amorchewin and Worapoj Kreesuradej proposed and algorithm called Promising Frequent Itemset Algorithm. An itemset is called promising that satisfies the following equation:

$$min\_sup_{(DB)}-\left(\frac{MAXSUPP}{TOTALSIZE}\right)*inc\_size<=min\_PL<min\_supDB \quad (4)$$

Where $min\_sup_{(DB)}$ is minimum support count for an original database, maxsupp is maximum support count of itemsets, total size is a number of transaction of an original database and inc_size is a maximum number of new transaction [11].

In this paper, apriori algorithm is applied to find all possible frequent k-itemsets and promising frequent k-itemsets. Apriori scans all transactions of an original database for each iteration with 2 steps process are join and prune step. Unlike typical apriori algorithm, items in both frequent k-itemsets and promising frequent k-itemsets can be joined together in the join step. For a frequent item, its support count must be higher than a user-specified minimum support count threshold and for a promising frequent item, its support count must be higher than min_PL but less than the user-specified minimum support count [11].

Table 11 Transaction Data

TID	List of Item
1	A,B,E
2	B,D
3	B,C
4	A,B,D
5	A,C
6	B,C
7	A,C
8	A,B,C,E
9	A,B,E
10	A,C

Table 12 Candidate 1-itemset

Itemset	Support
A	7
B	7
C	6
D	2
E	3

Table 13 Candidate, frequent and promise frequent 2-itemset

C2	support
AB	4
AC	4
AD	1
AE	3
BC	3
BD	2
BE	3
CD	0
CE	1
DE	0

L2	support

AB	4
AC	4
PL2	support
AE	3
BC	3
BD	2
BE	3

As an example, an original database shown in figure has 10 transactions, i.e. |DB| = 10. Then 3 new transaction is inserted into original database i.e. |db| = 3. Here minimum support count for mining association rules is set to 4. So, min_PL is computed as : min_PL = $4-(\frac{7}{10})*3 = 2$.

Therefore, if any itemset has support count at least 2 but less than 4, then it will be promising frequent itemset[11].

This algorithm works better even with incremental database and find closed/promising frequent itemset efficiently. The future work that can be done is modifying apriori approach for finding frequent itemset.

III. CONCLUSION

The promising frequent itemset algorithm is much better algorithm than any other algorithm presented here as this algorithm can work efficiently with incremental database and also find frequent and promising frequent itemset efficiently.

ALGORITHM	REVIEW
Apriori	Frequent rescanning of database. Not suitable for incremental approach
Improved Apriori	Reduce the transaction better than apriori. But not helpful in increasing database handling
CTFI and BCTFI	Binary to decimal and vice versa conversion become complex when there is a huge data base. So efficiency may go down.
MRFI-del	Restore the closed itemset after delete Transction
IMS Apriori	Difficult to handle incremental approach due to not considering any parameter for increasing database
Promising Frequent Itemset Algorithm	Efficiently find frequent and promising frequent itemset.Basically use the apriori algorithm for frequent itemset.So much better modification in apriori can be done.

REFERENCES

[1] Agarwal R , Ramakrishnan Srikant, " Fast algorithms for mining association rules" Proc. 20th int. conf. very large data bases, VLDB, 1994.

[2] A.M.J. Md. Zubair Rahman and P. Balasubramanie,"Weighted Support Association Rule Mining using Closed Itemset Lattices in Parallel", IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.3, March 2009

[3] Christian Borgelt, Xiaoyuan Yang, Ruben Nogales-Cadenas, Pedro Carmona-Saez, Alberto Pascual-Montano, "Finding Closed Frequent Item Sets by Intersecting

[4] Dr. S.S. Mantha, Madhuri Rao, Ashwini Anil Mane, Anil S. Mane," Mining Maximal Frequent Item Sets", International Journal of Computer Applications (0975 – 8887) Volume 10– No.3, November 2010

[5] Daniel Kunkle, Donghui Zhang, and Gene Cooperman Daniel Kunkle, Donghui Zhang,Gene Cooperman," Mining Frequent Generalized Itemsets and Generalized Association Rules Without Redundancy", JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 23(1): 77{102 Jan. 2008

[6] Jaishree Singh, Hari Ram, Dr. J.S. Sodhi ,"Improving Effficiency of Apriori Algorithm Using Transaction Reduction", International Journal of Scientific and Research Publications, Volume 3, Issue 1, January 2013

[7] Lun Tan, Yingyong Bu, Haiming Zhao," Incremental Maintenance of Association Rules Over Data Streams", 2010 International Conference on Networking and Digital Society

[8] Laszlo Szathmary, Amedeo Napoli, Petko Valtchev," Towards Rare Itemset Mining",Tools with Artificial Intellegence,2007.

[9] Maragatham G, Lakshmi M," A RECENT REVIEW ON ASSOCIATION RULE MINING", Indian Journal of Computer Science and Engineering (IJCSE)

[10] Marghny H. Mohamed • Mohammed M. Darwieesh," Efficient mining frequent itemsets algorithms", Springer-Verlag Berlin Heidelberg 2013

[11] Ratchadaporn Amornchewin, Worapoj Kreesuradej, "Incremental Association Rule Mining Using Promising Frequent Itemset Algorithm", ICICS 2007

[12] R. Uday kiran, P Krishna Reddy," An Improved Multiple Minimum Support Based Approach to Mine Rare Association Rules", *2009 IEEE Symposium on Computational Intelligence and Data Mining (IEEE CIDM 2009)*

[13] Show-Jane Yen, Yue-Shi Lee, Chiu-Kuang Wang, "The Maintenance Of Representative Frequent Itemsets", Proceedings of the Ninth International Conference on Machine Learning and Cybernetics, Qingdao, 11-14 July 2010

[14] Sotiris Kotsiantis, Dimitris Kanellopoulos," Association Rules Mining: A Recent Overview", GESTS International Transactions on Computer Science and Engineering, Vol.32 (1), 2006, pp. 71-82

[15] Thomas Bernecker, Hans-Peter Kriegel, Matthias Renz, Florian Verhein, Andreas Zuefle, "Probabilistic Frequent Itemset Mining in Uncertain Databases", Proc. 15th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining (KDD'09), Paris, France, 2009

[16] Tzung-Pei Hong, Chun-Wei Lin, Yu-Lung Wu, "Incrementally fast updated frequent pattern trees", Expert Systems with Applications 34 (2008) 2424–2435

[17] Tianming Hu, Sam Yuan Sung, Hui Xiong, Qian Fu," Discovery of maximum length frequent itemsets", *Journal of Information Sciences 4 February 2007*

[18] Wang, L; Cheung, DWL; Cheng, R; Lee, SD; Yang, XS "Efficient mining of frequent item sets on large uncertain data", IEEE Transactions on Knowledge & Data Engineering, 2012, v. 24 n. 12, p. 2170-2183 databases

[19] XindongWu, Vipin Kumar,J. Ross Quinlan, Joydeep Ghosh, Qiang Yang ,Hiroshi Motoda, Geoffrey J. McLachlan , Angus Ng, Bing Liu, Philip S. Yu , Zhi-Hua Zhou, Michael Steinbach, David J. Hand, Dan Steinberg, "Top 10 algorithms in data mining", Springer-Verlag London Limited 2007