International Journal of Engineering & Technology, 7 (3.34) (2018) 309-312

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Hash based Approach for Mining Frequent Item Sets from Transactional Databases

[1]UMohan Srinivas, [2]Ch Anuradha, [3]Dr. P. Sri Rama Chandra Murty

1Associate Professor, Dept. of Computer Science & Engineering, QIS College of Engineering and Technology, Ongole, Andhra Pradesh,India.
2Asst. Professor, Dept. of Computer Science & Engineering, V. R. Siddhartha Engineering College, Vijayawada, Andhra Pradesh,India.
3Asst. Professor, Dept. of Computer Science & Engineering, University College of Engineering & Technology, Acharya Nagarjuna University, Guntur, Andhra Pradesh, India
Corresponding author E-mail: umohansrinivas@gmail.com anuradha.chinta4@gmail.com chandra_psr@rediffmail.com

Abstract

Frequent Itemset Mining become so popular in extracting hidden patterns from transactional databases. Among the several approaches, Apriori algorithm is known to be a basic approach which follows candidate generate and test based strategy. Although it is efficient level-wise approach, it has two limitations, (i) several passes are required to check the support of candidate itemsets. (ii) Towards more candidate itemsets and minimum threshold variations. A novel approach is proposed to tackle the above limitations. The proposed approach is one pass Hash-based Frequent Itemset Mining to derive frequent patterns. HFIM has feature that maintains candidate itemsets dynamically which are independent on minimum threshold. This feature allows to limit the number of scans over the database to one. In this paper, HFIM is compared with the Apriori to show the performance on standard datasets. The result section shows that HFIM outperforms Apriori over large databases.

Keywords: *Frequent Itemset Mining, Apriori Algorithm, minimum threshold.*

1. Introduction

Frequent Itemset Mining become so popular data mining technique in extracting hidden information that is highly correlated itemsets from a large transactional database. Transactional database is defined as follows. Let DB be a set of tuples [1]{T1, T2, T3, …., Tm}, {I1, I2, I3, …., In} be a set of distinct items. An itemset X is set of items which is a subset of I $X \leq I$. The support of an itemset is determined from the occurrence of itemset X in the transactions with respect to the total number of transactions in the database. The itemset X is said to be frequent if its support reaches the given user threshold value. So the goal of FIM is to derive such kind of frequent itemsets from database.
FIM approaches have been classified into two categories, the first category, approaches are based on Apriori Heuristic [1]. Here they first generate candidates size k from its frequent itemsets of size (k-1) length in a level-wise manner and then find its support by scanning the database. In the second category, approaches are based on the FPgrowthheuristic [2]. They represent the entire database as a compressed tree in main memory, then visit tree recursively to find all frequent patterns. Although second classifications are efficient than aprioriin terms of number of passes and candidates, they consume high amount of memory space especially for the large databases.
In this paper, we propose a Novel approach called HFIM (Hash-based Frequent Itemset Mining), which derived frequent itemsets with a single scan of the database DB. Here, the candidate itemsets are generated and maintained in hash table per each transaction with their support. Candidate itemsets are incremented by one if it is already maintained in the hash table, when reading new transaction. If the itemset does not exist, one new entry is created in the hashing table and set to 1. At the end of database, each itemset frequency is examined and compared with the threshold to declare frequent itemsets whose support is greater than or equal to threshold. In the result section, results of the proposed approach HFIM is presented and it outperforms Apriori approach for the variety of databases.
The rest of the paper is organized as follows. The next section2 reviews the existing techniques of FIM. In the next section3, the proposed HFIM is presented. The performance of the proposed approach is presented in the section 4, and conclusions are presented in the last section.

2. Related Work

FIM approaches can be classified into two categories. The first classification is the level-wise candidate generate and test based strategy, where the candidate itemsets are generated first and then tested their support against the database. The second classification follows divide and conquer policy, where compressed tree is used to maintain the entire database and recursively visited to derive all the frequent itemsets. The detailed explanation for these approaches are given below.
The popular approach under generate and test strategy is Apriori proposed by Aqrawal et al [1]. In this approach, candidates are generated in a level-wise manner recursively from its previous length frequent information. The candidate generation algorithm

generates k-size candidates from (k-1) sized itemsets. Then those candidates are validated and incremented by 1 against the database. This process is repeated until no more candidates are generated.

Let us consider the transaction database illustrated in Table 1[1], itrecorded with 9 transactions and 5 items{I1, I2, I3, I4 and I5}.

Table 1: Transactional Database

Transaction ID	List of item ids
1	I1, I2, I5
2	I2, I4
3	I2, I3
4	I1, I2, I4
5	I1, I3
6	I2, I3
7	I1, I3
8	I1, I2, I3, I5
9	I1, I2, I3

Fig 1 shows list of candidate and frequent itemsets of the Apriori algorithm for a given user threshold 22% or mincount =2. Database D is visited to calculate the frequency/support of k-candidate itemsets (k=1), then 1-length itemsets are extracted and presented in fig 1. For the example of Table1, all items are considered as candidates and frequent since their support reached threshold (mincount=2). In second iteration, 2-candidate itemsets are generated from 1-length frequent itemsets (simple join). Database D is scanned to calculate the support of 2-candidate itemsets and 2-length itemsets are derived as {I1 I2, I1 I3, I1 I5, I2 I3, I2 I4, I2 I5}.{ I1 I2 I5, I1 I2 I3} are candidates generated by joining 2-length frequent items as size of 3. Fortunately all of them are identified as frequent because their support is 2. This process is terminated because no candidates are generated for the size of 4. Finally the frequent itemsets for a given threshold 0.22, { I1, I2, I3, I4, I5, I1 I2, I1 I3, I1 I5, I2 I3, I2 I4, I2 I5, I1 I2 I5, I1 I2 I3}.

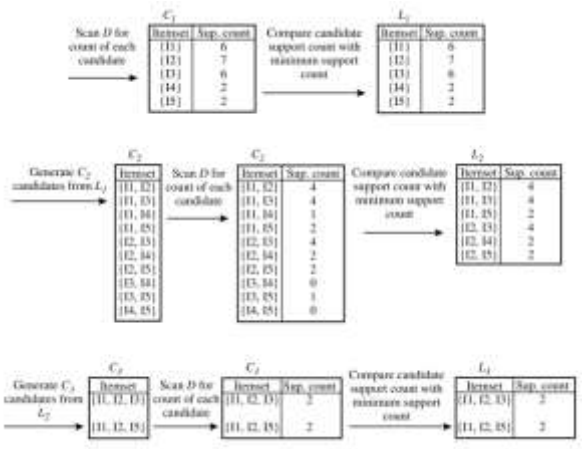

Figure1: Apriori Algorithm Illustration for Table 1[1]

Limitations of Apriori [15]:

Too many scans on transaction database DB is to be required: To find the cumulative support of itemsets. For each iteration, all existing Aprioribased approaches need to scan once a database. Thus, number of passes/scans proportional to the length of candidate itemset (Maximum length).

Low Minimum threshold: these approaches biased to the minimum threshold, because when low threshold value is considered, more number of candidates are required to process. Hence more computation time and scans are necessary to process candidates.

The features of Apriori (downward closure and antimonotic properties) are attracted by many researchers and result with many approaches. The one is, DIC (Dynamic Itemset Counting) proposed by Brinet et al. [3]. In DIC, where the database is divided into equal partitions such a way that it fit into the memory. It collects information about itemsets from the first partitions, then classify them as strong, small, suspected small, suspected large itemsets. And locally found large or frequent itemsets are used to generate the next length candidates. Then second partition is read to validate the maintained candidate itemsets. If the itemsets are moving from suspect state to strong state such itemset combinations are pushed to the candidate itemsets. This process is repeated until candidate itemsets size become to zero. Muller et al.[4] has proposed prefix tree based sequential FIM. It is similar to Apriori, but prefix structure allows faster performance. However, the prefix structure required to keep both frequent and candidate itemsets, which causes for the performance degrade with respect to the memory usage. Eclat algorithm proposed by Zaki et al. [5] uses vertical list of transaction ids as tidlist. Itemsets are categorized into disjoint equivalence classes w.r.t their common (k-1) prefixes. K-length or K+1 candidates are generated from its (k-1) or k length itemsets equivalence classes. The support of itemset is computed from the intersection of their tidlist. Same author has presented [6] variant, a data structure is proposed which maintains tidlist of itemset which are not presented to overcome the size of tidlist. But it supports only for deriving regular frequent patterns.

Second one divide and conquer starts with the first and popular algorithm FPgrowth, uses compressed tree FP tree for deriving itemsets without generating candidates. The algorithm is comprised in two phases: (i) it constructs tree FP by mapping each transaction as each path in the tree, where each node maintains its support. (ii) Derives all the frequent patterns by visiting FP tree in bottom up in a recursive manner. These kind of approaches shows faster response, but limited to the smaller databases since more memory space is required to keep active in the computation area. The NFPgrowth algorithm proposed by Cerf et al. [8] improves the performance of FPgrowth by maintaining frequent patterns tree as independent table, which improves the speed and allows to create tree only once. Variant of FPgrowth algorithm was proposed [7] to deal uncertain data. FP-array techniques was proposed [9,10] to reduce the traversing time of FP tree, which aims at frequent itemsets as well as other patterns like categorical, maximal and close itemsets.

The above classification, first one takes more passes and candidates to process database for frequent itemsets, whereas second one is proposed to avoid candidate itemset generation and less passes. However, second one follows divide and conquer strategy and recursive mining procedure allows to consuming huge memory space. These kind of approaches need to be considered for the improvement, since the current transactional databases are very large with million records [11]. Recently, bio inspired approaches has been added to the above two classification to reduce competency over the database. BSO-ARM[12], PGARM[13] and PeARM[14] are the few considered as review among them. These king of bio inspired approaches are limited to the regular frequent itemsets only.

3. Hash Bashed Frequent Itemset Mining: (Hfim):

This section discusses the proposed method Hash based frequent itemset mining. The algorithm illustration immediately followed by an analysis of HFIM compared with Apriori.

3.1 HFIM Algorithm:

The main aim of the proposed approach HFIM is to reduce the number of scans over the database DB. The basic idea is to generate or maintain all possible candidate itemsets for each transaction Ti from DB in Hash table. While processing new

transaction, support count of itemsets of hash table is incremented by 1, if such itemset is already generated and maintained in hash table.Otherwise Itemset support is initialized to 1, if it is not generated previously. The same procedure is repeated until all the transactions of DB have been visited.
The HFIM takes one scan of the DB to derive all frequent itemsets, it can be said to *complete*. Because frequent itemsets are directly derived from the transactions of the DB whose support is greater than or equal to ***m*minsup***.

Algorithm 1. HFIM Algorithm
Input: DB: Transactional database. Minsup: minium support threshold. Output: FI: All Frequent Itemsets

for each transaction T_j do
C<- GenerateCandidates(T_j).
For each itemseti∈C
If i∈Hash then
Hash(t)---Hash(t)+1
else Hash(t)←1
end for
end for
FI←∅
for itemset s ∈Hash do
If Hash(s) ≥minsup then
FI← F U s.
end if
end for
return FI.

3.2 HFIM Algorithm Description:

HFIM consider transactional database DB as input, and the minimum threshold as minsup. It uses additional storage datastructure Hash table represented by Hash to store and manipulate all the itemsets. Finally algorithm returns the set of itemsets whose support >= minsup.
At first, candidate itemset C is generated from each transaction T_j. for example, transaction 1 contains I1, I2 and I5. And the possible candidate itemsets are { I1, I2, I5, I1 I2, I1 I5, I2 I5, I1 I2 I5}. After that, each itemseti∈Cis stored the hash table Hash. Hash is initialized to empty, so entries are created with the item name as keys and the candidates support is initialized to 1. Otherwise, means if it is matching with the key then its support will be incremented by 1. After visiting all the transactions, FI is returned with the list of itemsets whose support reaches minsup. The visual representation of HFIM approach for the Table1 with minsup=2 is presented in fig 2. Fig 2 shows that the frequent itemsets of the example is same as apriori shown in fig 1.

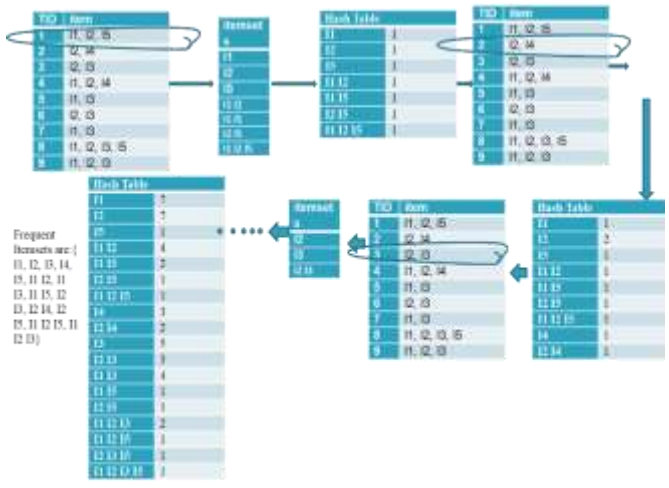

Figure 2: Illustration of HFIM for Table 1.

4. Experimental Results:

Several experiments have been presented to show the performance of HFIM with the standard data sets used in [16]. The first dataset is recorded with transactions of 2178, number of items from 8 to 16, and the average of items per transaction is between 8 and 16.
The second data set with medium sized database size of 4 with transactions size varies from 59000 to 100000, number of items from 500 to 16000, average items is from 2 to 10.
Third dataset is a large database BMPPOS, which is recorded with 500000 transactions and above 1660 items with average items 2.5.
All the experiments are carried with the configurations of Intel I3 processor, 4GB RAM, and implemented in JAVA.

Table 2: Run time comparison of HFIM and Apriori

Data set Name	HFIM	Apriori
Bolts	140	4
Sleep	110	6
Pollution	821	20
Basket ball	18	15
Quake	50	29
BMS-WebView-1	45	1002
BMS-WebView-2	80	3985
retail	525	4895
Connect	1285	2600
BMP POS	500	9825

Table 2 is presented with runtime comparison of HFIM and Apriori using standard datasets. It is observed that Apriori approach is performing better than Apriori. However for medium and large database, HFIM outperforms Apriori more than twice. In other way, we can say that HFIM is efficient than apriori when the database is dence.
The second experimental result is shown in Fig 3. It is observed that HFIM outperforms Apriori when the minsup is getting decreased.

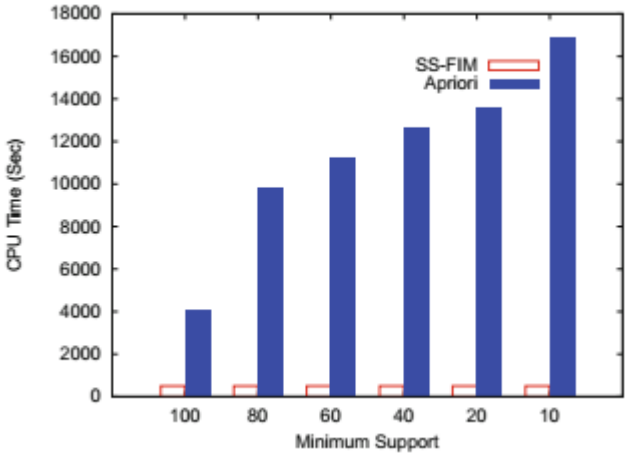

Figure 3: Runtime Comparison of HFIM and Apriori w.r.t minsup

5. Conclusion:

This paper has proposed HFIM approach, it is able to extract frequent itemsets with only one scan. Candidate itemsets and

intermediate information is stored in a data structure Hash Table, which is helped in reducing scans to one. Experimental results are showing that the proposed method is performing better than Apriori, which maintains all the candidates in a level wise manner that causes for more scans.

The proposed method is only compared with Apriori since the goal is to reduce the scans over the database. However, it is required to be modified to outperform approaches like DIC to reduce the number of candidates.

It is also observed that HFIM can be extended to handle Big data specified data like sensor data and event log data.

References:

[1] Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets ofitems in large databases. In: ACM SIGMOD Record, vol. 22, no. 2, pp. 207–216.ACM, June 1993

[2] Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation.In: ACM SIGMOD Record, vol. 29, no. 2, pp. 1–12. ACM, May 2000

[3] Brin, S., Motwani, R., Ullman, J.D., Tsur, S.: Dynamic itemset counting and implication rules for market basket data. In: ACM SIGMOD Record, vol. 26, no. 2, pp.255–264. ACM, June 1997

[4] Mueller, A.: Fast sequential and parallel algorithms for association rule mining: acomparison. Technical report CS-TR-3515, University of Maryland, College Park,August 1995.

[5] Zaki, M.J., Parthasarathy, S., Ogihara, M., Li, W.: New algorithms for fast discovery of association rules. In: Third International Conference Knowledge Discoveryand Data Mining (1997).

[6] Amphawan, K., Lenca, P., Surarerks, A.: Efficient mining top-k regular-frequentitemset using compressed tidsets. In: Cao, L., Huang, J.Z., Bailey, J., Koh, Y.S.,Luo, J. (eds.) PAKDD 2011. LNCS (LNAI), vol. 7104, pp. 124–135. Springer,Heidelberg (2012). doi:10.1007/978-3-642-28320-8 11.

[7] Leung, C.K.-S., Mateo, M.A.F., Brajczuk, D.A.: A tree-based approach for frequent pattern mining from uncertain data. In: Washio, T., Suzuki, E., Ting, K.M.,Inokuchi, A. (eds.) PAKDD 2008. LNCS (LNAI), vol. 5012, pp. 653–661. Springer,Heidelberg (2008). doi:10.1007/978-3-540-68125-0 61.

[8] Cerf, L., Besson, J., Robardet, C., Boulicaut, J.F.: Closed patterns meet n-aryrelations. ACM Trans. Knowl. Discov. Data (TKDD) 3(1), 3 (2009).

[9] Grahne, G., Zhu, J.: Fast algorithms for frequent itemset mining using FP-trees.IEEE Trans. Knowl. Data Eng. 17(10), 1347–1362 (2005).

[10] Borgelt, C.: Frequent itemset mining. Wiley Interdisc. Rev.: Data Min. Knowl.Discov. 2(6), 437–456 (2012).

[11] Djenouri, Y., Bendjoudi, A., Mehdi, M., Nouali-Taboudjemat, N., Habbas, Z.:GPU-based bees swarm optimization for association rules mining. J. Supercomput. 71(4), 1318–1344 (2015).

[12] Djenouri, Y., Drias, H., Habbas, Z.: Bees swarm optimisation using multiple strategies for association rule mining. Int. J. Bio-Inspired Comput. 6(4), 239–249 (2014).

[13] Gheraibia, Y., Moussaoui, A., Djenouri, Y., Kabir, S., Yin, P.Y.: Penguins searchoptimisation algorithm for association rules mining. CIT J. Comput. Inf. Technol.24(2), 165–179 (2016).

[14] Luna, J.M., Pechenizkiy, M., Ventura, S.: Mining exceptional relationships withgrammar-guided genetic programming. Knowl. Inf. Syst. 47(3), 571–594 (2016).

[15] Hegland, M.: The apriori algorithm tutorial. Math. Comput. imaging Sci. Inf.Process. 11, 209–262 (2005).

[16] Guvenir, H.A., Uysal, I.: Bilkent university function approximation repository(2000). http://funapp.CS.bilkent.edu.tr/DataSets. Accessed 12 Mar 2012.