ACSIJ Advances in Computer Science: an International Journal, Vol. 3, Issue 5, No.11 , September 2014
ISSN : 2322-5157
www.ACSIJ.org

ACSIJ
WWW.ACSIJ.ORG

CBT-fi: Compact BitTable Approach for Mining Frequent Itemsets

A.Saleem Raja[1] and E.George Dharma Prakash Raj[2]

[1]Research Scholar, Department of Computer Science, Engineering and Technology, Bharathidasan University, Trichy, Tamil Nadu, India.
Email: asaleemrajasec@gmail.com

[2]Assistant Professor, Department of Computer Science, Engineering and Technology, Bharathidasan University, Trichy, Tamil Nadu, India.

Abstract

Frequent item-set mining is a data analysis method which is used to find the relationship between the different items in the given database. Plenty of research work and progress has been made over the decades due to its wider applications. Recently, BitTableFI and Index-BitTableFI approaches have been applied for mining frequent item-sets and results are significant. They use Bit Table as the base data structure and exploits the bit table both horizontally and vertically. However still needs simple and efficient approach for mining frequent itemsets from the given dataset. This paper introduces the Compact BitTable approach for mining frequent itemsets (CBT-fi) which clusters(groups) the similar transaction into one and forms a compact bit-table structure which reduces the memory consumption as well as frequency of checking the itemsets in the redundant transaction. Finally we present result, which shows the proposed algorithm has better than the existing algorithms.

Keywords: *Frequent Itemset Mining, Bit-Table, Association Rule Mining, BitTableFI*

1. Introduction

Goal of the data mining is to discover potentially useful information embedded in databases. Association rule mining is the one of the data mining technique which was introduced in 1993[2]. It finds the interesting association and/or correlation relationships among large set of data items [9]. Mining frequent itemset is the primary task in mining association rules. A typical and widely used example of frequent item-sets mining is to analyze supermarket transaction data, that is, to examine customer behavior in terms of the purchased products. Frequent sets of products describe how often items are purchased together. In addition to this frequent itemset mining have applications in areas such as bioinformatics, fraud detection and web usage mining [5]. Many algorithms have been proposed to find frequent item-sets. They can be grouped into following categories [7,8].

a) Candidate generation and test approach. Example: Apriori[2] and BitTableFI [3]
b) Pattern growth approach. Example: FP-growth[4]
c) Hybrid approach. Example: Eclat[10] and Index-BitTableFI[6].

Even though many algorithms have been proposed recent years, FI mining is remains challenging task due its complexity. Therefore simple and computationally efficient algorithms are desirable. This paper introduces CBT-fi, which uses simple and efficient data structure called compact BitTable for storing clustered transaction. The compact BitTable contains only unique transactions with record-count-vector (rcv)and bit-count-vector(bcv) used to find the frequent itemsets with less number of iterations.

The rest of the paper is organized as follows. Section 2 presents related works. The proposed algorithm and example of this algorithm in section 3 and Section 4 presents the result of experiments. Finally we conclude the paper.

2. Related Work

The Apriori[2], FP-growth[4] algorithms are the base algorithms for many latest FI mining algorithms. Apriori uses an efficient candidate generation method such that each level uses the candidate itemsets which are generated in its previous level. However it requires multiple database scanning for generating FI. FP-growth is a representative pattern growth approach. It is a Depth First Approach (DFS) and uses a special data structure, FP-Tree, for compact representation of the original database. Only two database scans are needed for the algorithm and no candidate generation is required. This makes the FP-growth method much faster than Apriori. But FP-tree construction for large dataset become complex. Many research works has been made over the decades to improve the efficiency of FI mining.

Recently Dong and Han proposed an algorithm named as BitTableFI [3]. In the algorithm, a special data structure BitTable is used horizontally and vertically to compress database for quick candidate item-sets generation and support count, respectively. But the BitTableFI suffers from the high cost of candidate generation and test.

Song et al, proposed a new algorithm Index-BitTableFI[6]. It also uses BitTable horizontally and vertically. To make use of BitTable horizontally, index array and the corresponding computing method are proposed. By computing the subsume index, those itemsets that co-occurrence with representative item can be identified quickly by using breadth-first search at one time. Then, for the resulting itemsets generated through the index array, depth-first search strategy is used to generate all other frequent itemsets. However, Index-BitTableFI always uses a fixed size of Bit-Vector for each item (equal to number of transactions in a database). It leads to consume more memory for storage Bit-Vectors and the time for computing the intersection among bit-vectors [7,8].

Janos proposed a novel algorithm [1] based on BitTable (or bitmap) representation of the data. Data - related to frequent item-sets - are stored in spare matrices. Simple matrix and vector multiplications are used to calculate the support of the potential n+ 1 item-set. Even though novel bitmap-based approach is simple but involves more matrix multiplications which lead to increase the computing.

Vo et al, proposed the dynamic bit vectors [7] algorithm for constructing a DBV tree and mining FIs from a database. This algorithm shows the better performance result but still it involves computation complexity by constructing DBV tree.

3. Proposed Algorithm

This section presents the CBT-fi, which uses simple and efficient data structure called compact BitTable for storing clustered transaction. The compact BitTable contains only unique transactions with record-count-vector (rcv) and bit-count-vector (bcv) used to find the frequent itemsets with less number of iterations. CBT-fi approach has two major parts. 1. Computing compact BitTable with record-count-vector and bit-count-vector 2. Generate the frequent itemset using compact BitTable. Initially we start with problem statement.

3.1 Problem Statement

The problem of mining frequent item-sets is formally stated by definitions 1-3 and lemma 1.
Frequent item-sets mining is defined as follows:

Let $T= \{t_1, \ldots, t_n\}$ be the set of transaction in the database D and let $I=\{i_1, \ldots, i_m\}$ be the set of items and each transaction can be identified by a distinct identifier tid.
Definition 1: A set $X \in I$ is called an itemset. An itemset with k items is called a k-itemset.
Definition 2: The support of an item-set *X*, denoted as *sup(X)*, is defined as the number of transactions in which *X* occurs as a subset.
Definition 3: For a given D, let min_sup be the threshold minimum support value specified by user. If $sup(X) \geq min\_sup$, item-set X is called a frequent item-set.
The task FIM is to generate all frequent item-sets in the database, which have a support greater than min_sup.
Lemma 1: A subset of any frequent item-sets is a frequent item-set, a superset of any infrequent itemset is not a frequent item-set.

3.2 CBT-fi Algorithm

The major components the BitTable (BitMap or Matrix), which is efficient data structure for mining frequent item-sets [7,8,9,10]. The process begins with, the transaction database can be transformed into a binary matrix M_1, in which each row corresponds to a transaction and each column corresponds to an item. Therefore the bit-table contains 1 if the item is present in the current transaction and 0 otherwise.

$M(i,j) = \begin{cases} 1 \\ 0 \end{cases}$ where M represents 2D matrix , i represents row and j represents item (col).

Once the M is formed, compute the column wise bit count for each item and eliminate the items column whose bit count is less than min_sup value. Consider an example database shown in Table 1.

Table 1: The example database

TID	Items
1	*A B C E F O*
2	*A C G*
3	*E I*
4	*A C D E G*
5	*A C E G L*
6	*E J*
7	*A B C E F P*
8	*A C D*
9	*A C E G M*
10	*A C E G N*

There are 14 different items and the database consists of 10 transactions. Read each transaction from the given database and form a bit table M as shown in figure 1 and eliminate the item's column whose bit count is less than min_sup. Assume min_sup=2 as shown in figure 2.

M	A	B	C	D	E	F	G	I	J	L	M	N	O	P
1	1	1	1	0	1	1	0	0	0	0	0	0	1	0
2	1	0	1	0	0	0	1	0	0	0	0	0	0	0
3	0	0	0	0	1	0	0	1	0	0	0	0	0	0
4	1	0	1	1	1	0	1	0	0	0	0	0	0	0
5	1	0	1	0	1	0	1	0	0	1	0	0	0	0
6	0	0	0	0	1	0	0	0	1	0	0	0	0	0
7	1	1	1	0	1	1	0	0	0	0	0	0	0	1
8	1	0	1	1	0	0	0	0	0	0	0	0	0	0
9	1	0	1	0	1	0	1	0	0	0	1	0	0	0
10	1	0	1	0	1	0	1	0	0	0	0	1	0	0
	8	2	8	2	8	2	5	1	1	1	1	1	1	1

Figure 1. Bit table representation

M	A	B	C	D	E	F	G
1	1	1	1	0	1	1	0
2	1	0	1	0	0	0	1
3	0	0	0	0	1	0	0
4	1	0	1	1	1	0	1
5	1	0	1	0	1	0	1
6	0	0	0	0	1	0	0
7	1	1	1	0	1	1	0
8	1	0	1	1	0	0	0
9	1	0	1	0	1	0	1
10	1	0	1	0	1	0	1
	8	2	8	2	8	2	5

Figure 2. Frequent single items

Once we form the frequent single items, the next step is to sort the frequent single items (A B C D E F G) in ascending order (B D F G A C E) based on the support count. If two items have the same supports, they will be sorted according to lexicographic order, as shown in below figure 3.

M	B	D	F	G	A	C	E
1	1	0	1	0	1	1	1
2	0	0	0	1	1	1	0
3	0	0	0	0	0	0	1
4	0	1	0	1	1	1	1
5	0	0	0	1	1	1	1
6	0	0	0	0	0	0	1
7	1	0	1	0	1	1	1
8	0	1	0	0	1	1	0
9	0	0	0	1	1	1	1
10	0	0	0	1	1	1	1

Figure 3. Frequent single items in ascending order.

After sorting the frequent single items in ascending order, next steps is to the cluster(group) the similar transaction (row) based on the decimal value of each row is denoted as record-count vector(rcv) and also compute the bit count for each transaction(row) is denoted as bit-count-vector(bcv) as shown below figure 4.

B	D	F	G	A	C	E	rcv	bcv
1	0	1	0	1	1	1	2	5
0	0	0	1	1	1	0	1	3
0	0	0	0	0	0	1	2	1
0	1	0	1	1	1	1	1	5
0	0	0	1	1	1	1	3	4
0	1	0	0	1	1	0	1	3

Figure 4. Compact Bit Table with rcv and bcv.

Out of 10 transactions with 14 items, the compact bit table contains 6 transactions with 7 items. The example shows that compact bit table saves the memory space and will reduces number of iterations involved in FI mining considerably. The pseudo code for generating compact bit table is shown in algorithm 1.

Algorithm 1. The pseudo code of Compact Bit Table algorithm

Scan database D once, store the bit value in M and Delete infrequent items based on the min_sup
Sort frequent single items in ascending order based on support count
Count(C) the similar transactions in M ,
keep unique transaction in CBT and store the count value(C) in rcv
for each transaction in CBT do begin
count number of 1 in each transaction
store it in cbv
end
Delete M and write CBT, rcv and bcv

Compact bit table will be used for further FI mining process. Based on the frequent single item-set, generate the candidate 2-item-sets and compute the support count for each candidate 2-item-set. If the support count is greater than min_sup then that item-set is added to the frequent 2 item-set. Based on the frequent 2 item-set, generate the candidate 3-item-sets and compute the support count for each candidate 3-item-set. If the support count is greater than min_sup then that item-set is added to the frequent 3 item-set. This process will be repeated till final FI is generated The pseudo code for generating FI is shown in algorithm 2.

Algorithm 2. The pseudo code of FI mining algorithm

C_k: candidate item-set of size k
L_k: frequent item-set of size k
L_1= {frequent items}
for (k=1; Lk ≠ Ø; k++) do begin

C_{k+1}=candidates generated from L_k
for each candidate item-set CI in C_{k+1} do begin
sc=0
for each transaction t in CBT whose bcv ≥ CI.size do begin
find the column position of each element in CI and check for bit value 1 in t.
if bit value for all elements position = =1 then
sc=sc+ rcv[t]
end if
if sc>min_sup then
L_{k+1}=CI
end if
end
end

Consider the above example, 2-itemsets which are satisfy the min_sup are {BF:2, BA:2, BC:2, BE:2, DA:2, DC:2, FA:2, FC:2, FE:2, GA:5, GC:5, GE:4, AC:8, AE:6, CE:6}. These items set will consider for finding frequent 3- items set. 3-itemsets which are satisfy the min_sup are {BFA:2, BFC:2, BFE:2, BAC:2, BAE:2, BCE:2, DAC:2, FAC:2, FAE:2, FCE:2, GAE:4, GCE:4, GAC:5, ACE:6 }. These items set will consider for finding frequent 4- items set. 4-itemsets which are satisfy the min_sup are {BFAC:2, BFAE:2, BFCE:2, BACE:2, FACE:2,GACE:4}. These items set will consider for finding frequent 5- items set. 5-itemsets which are satisfy the min_sup are {BFACE:2}.

Table 2: Features of the test database

Database	#Trans	#Items
Chess	3196	76
Accidents	340183	468

4. Experimental Result

Experiments were conducted to show the performance of the proposed algorithms. The algorithms were coded in Java in netbean framework. Two standard databases were used for the experiments, with their features displayed in table 2. Figure 5 shows the mining time of chess database. Figure 6 shows the mining time of accidents database. The results show that proposed approach is better than Index-BitTableFI and DBV-FI.

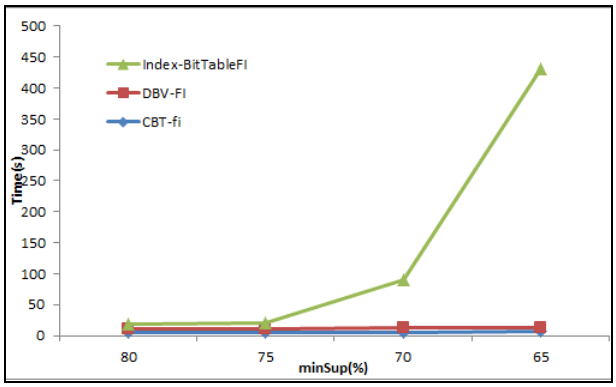

Figure 5. Execution time of the three algorithms for chess under different minSup values

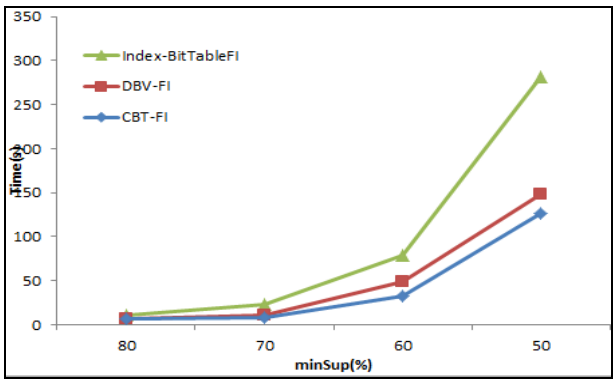

Figure 6. Execution time of the three algorithms for accidents under different minSup values.

5. Conclusions

In this paper, we proposed a new approach for mining frequent itemsets from transaction databases. Proposed approach uses bit-table as the base data structure and has two parts. First algorithm computes the CBT with rcv and bcv. The CBT saves the memory considerably by clustering the similar transactions. Second, it mines the FI from the CBT using rcv and bcv. The results show that proposed approach is better than Index-BitTableFI and DBV-FI.

References

[1] Abonyi J., "A Novel Bitmap-Based Algorithm for Frequent Itemsets Mining," Computational Intelligence in Engineering Studies in Computational Intelligence, vol. 313, pp. 171-180, 2010.

[2] Agrawal R, Srikant R., "Fast algorithms for mining association rules in large databases," *in Proceedings of the 20th International Conference on Very Large Data Bases*, San Francisco, USA, pp. 487–499, 1994.

[3] Dong J, Han M., "BitTableFI: An efficient mining frequent itemsets algorithm," Knowledge-Based Systems, vol.20 no.4, pp.329-335, 2007.
[4] Han J, Pei J, and Yin Y., "Mining frequent patterns without candidate generation," *in Proceedings of the ACM SIGMOD international conference on Management of data*, New York, USA , pp. 1-12, 2000.
[5] Schlegel B, Gemulla R, Lehner W., "Memory-Efficient Frequent-Itemset Mining," *in Proceedings of the 14th International Conference on Extending Database Technology*, New York, USA, pp. 461-472, 2011.
[6] Song W, Yang B, Xu Z., "Index-BitTableFI: An improved algorithm for mining frequent itemsets," Knowledge-Based Systems, vol.21, pp. 507–513. 2008.
[7] Vo B, Hong T.P, Le B, "Dynamic Bit Vectors: An efficient approach for mining frequent itemsets," Scientific Research and Essays, vol.6, pp.5358-5368, 2011.
[8] Vo B, Hong T.P, Le B, "DBV-Miner: A Dynamic Bit Vector Approach for fast mining frequent closed itemsets," Expert system with Applications, vol. 39, pp. 7196-7206, 2012.
[9] Yafi M, Al-Hegami A, Alam A and Biswas R., "YAMI: Incremental mining of Interesting Assocation Patterns," The International Arab Journal of Information Technology, vol.9, pp.504-510, 2012.
[10] Zaki M, Parthasarathy S, Ogihara M, Li W., "New algorithm for fast discovery of association rules," *in Proceeding of 3rd ACM SIGMOD international conference on Knowledge Discovery and Data Mining*, Menlo Park, USA, pp.283-286,1997.

A. Saleem Raja is from Tamil Nadu, India and doing P.hD in Bharathidasan University on the topic 'Agent based Distributed Data Mining Algorithm. He has completed bachelor's degree in Computer Science from Madurai Kamaraj University, Madutai and Masters Degree in Computer Applications from Anna university, Chennai. He also completed M.Phil from Periyar University, Salem and M.Tech from Bharathidasan University, Trchy. The author has attended national and international conferences and has written research papers on agent based distributed mining.

Dr. E. George Dharma Prakash Raj Completed his Masters Degree in Computer Science and Masters of Philosophy in Computer Science in the years 1990 and 1998. He has also completed his Doctorate in Computer Science in the year 2008. He has around twenty four years of Academic experience and thirteen years of Research experience in the field of Computer Science. Currently he is working as an Asst.Professor in the School of Computer Science and Engineering at Bharathidasan University, Trichy, India. He is an Editorial Board Member, Reviewer and International Programme Committee Member in many International Journals and Conferences. He has published several papers in International Journals and Conferences related to Computer Science He has convened many National and International Conferences related to Computer Science. His Areas of Interest are Computer Networks and Data Mining