IOSR Journal of Engineering (IOSRJEN) **www.iosrjen.org**
ISSN (e): 2250-3021, ISSN (p): 2278-8719
Vol. 09, Issue 5 (May. 2019), ||S (XI) || PP 81-84

Reduction of Candidate Set Generation for Association Rule Mining using Cardinality Count Technique

Shailendra Shende, Sachin Balvir,
Dept. of Information Technology YCCE, Nagpur
Dept. of Information Technology DMIETR, Nagpur

***Abstract*:** Since the discovery of large item set in a large database is a computationally expensive process, there are two ways to reduce the computational complexity of frequent itemset generation. The first one is by reducing the number of comparisons. Here instead of matching each candidate itemset against every transaction, we can reduce the number of comparisons by using more advanced data structures, either to store the candidate itemset or to compress the data set. Second approach is by reducing the number of candidate itemsets. The *Apriori* principle is an effective way to eliminate some of the candidate itemsets without counting their support values.
In this paper, we propose efficient algorithms to reduce the number of candidate itemsets for finding the frequent itemset, by enhancing the existing *Apriori* algorithm using static cardinality count method.
***Index Terms*:** *Association Rule Mining, Cardinality, Frequent Itemset.*

I. Introduction

Association rules are just one of the patterns that can be extracted from data by means of data mining techniques. Specifically, an association rule, X => Y, is a statement of the form "for a specified fraction of the total transactions, a particular value of the attribute set X determines the value of an attribute set Y with a certain confidence". In this sense, association rules aim to explain the presence of some attribute according to the presence or absence of some other attributes. The problem was studied first by Agrawal et al.[2]in 1993 on a supermarket basket data, an example of association rule is "In 20% of the transactions,90% of the people buying bread also buy butter in that transaction". Here, the support of the rule is 20%, and the confidence of the rule is 90%.

Because of the applicability and usefulness of association rules in many fields such as supermarket transaction analysis, telecommunications, word occurrence in text documents, user's visit to WWW pages(web usage mining), data analysis for earth sciences (to study ocean, land & atmospheric processes), stock exchange data analysis (for share market predictions), intrusion detection, bioinformatics & medical diagnosis etc. ,many researchers have proposed efficient algorithms to discover association rules[3,6,8,10,11]. The problem of discovering co-occurrence of items in a small data is a very simple task. However, the large volume of data makes this problem difficult and efficient algorithms are needed. In [2], the problem of discovering association rules is decomposed into two parts: Discovering all frequent itemset in the database, and generating the association rules from those frequent itemset. The second problem is a straightforward problem, and can be managed in polynomial time. On the other hand, the first task is difficult especially for large databases. The Apriori [1] is the first efficient algorithm on this issue, and many forthcoming algorithms are based on this algorithm.

Association Rule Mining (ARM) algorithms are computationally and I/O intensive. Hence main concern in ARM algorithms is reducing the computational cost. Computational cost of ARM algorithms can be reduced by following ways –

1. By reducing the number of passes over database.
2. By sampling the database
3. Through parallelization
4. By reducing the number of candidate itemsets generations

In this paper we propose the statistical approach of cardinality count to reduce the number of candidate itemsets generated in the intermediate steps and analyse the performance of proposed method with existing scheme on number of candidate itemset generated before and after pruning operation.

II. Problem Definition

An association rule in a transactional databases T, with I = $\{i_1, i_2, i_3,...,i_n\}$ be the set of all items, is an implication expression of the form X → Y, where X and Y are disjoint itemsets, i.e. , X ∩ Y = ϕ. The interestingness of an association rule can be measured in terms of its *support* and *confidence*. Support determines how often a rule is applicable to a given dataset, and confidence determines that how frequently items in Y appear in transactions that contain X. The formal definitions of these measures are:

Support,S(X=>Y) = σ(X*U*Y) / N

Confidence, C (X => Y) = σ(X*U*Y) / σ(X)

Where σ (X U Y), is a transaction containing both X and Y, N is total number of transactions and σ (X) is number of transactions containing X.

The problem of discovering association rule was first explored in [2] on supermarket basket data, that is the set of transactions that include items purchased by the customers. In this pioneering work, the data was considered to be binary, i.e. an item exists in a transaction or not, and the quantity of the item in the transaction is irrelevant.

In [1,2], mining of association rules was decomposed into two sub-problems: discovering all frequent patterns, and generating the association rules from those frequent itemsets. The second sub-problem is straightforward, and can be done efficiently in a reasonable time. However, the first sub problem is very tedious and computationally expensive for very large databases and this is the case for many real life applications. In large retailing data, the number of transactions is generally in the order of millions, and numbers of items (attributes) are generally in the order of thousands. When the data contains N items, then the number of possibly large itemset is 2N. However, the large itemset existing in the database are much smaller than 2N. Thus, brute force search techniques, which require exponential time, waste too much effort to obtain the set of large itemset. To reduce the number of possibly large itemsets, many efficient algorithms have been proposed. These algorithms generally use clever data structures (such as hash tables, hash trees, lattices, multi-hyper graphs, etc.) in order to reduce the size of possibly large itemsets and speedup the search process.

Generally, the efficiency of an association rule algorithm depends on the size of the candidate set (while generating and counting), and the number of scans over the database. As suggested in [4, 5, 7 , 9], most of the association rule algorithms concentrate on the following aspects to extract large itemsets efficiently:

1. Reducing I/O time by reducing the number of scans over the database,
2. minimizing the set of candidate itemsets,
3. counting the supports of candidate itemsets over the database in less time, and,
4. parallelizing the itemset generation.

In this sense, association rule algorithms generally differ on

1. The generation of the candidates,
2. counting the support of a candidate itemset,
3. number of scans over the database, and
4. The data structures employed.

III. Proposed Algorithm

In Apriori algorithm it is costly to handle a huge number of candidate sets. For example, if there are 10^4 frequent 1-itemsets, the Apriori algorithm will need to generate more than 10^7 length-2 candidates and accumulate and test their occurrence frequencies. Moreover, to discover a frequent pattern of size 100, such as[a1,….,a100], it must generate $2^{100} - 2 = 10^{30}$ candidates in total. This is the inherent cost of candidate generation.

In our following proposed algorithm we reduce this candidate generation cost, by using cardinality count statistical method as follows.

Algorithm static cardinality count

Input: set of transactions and minimum support (minsup).

Output: set of frequent itemsets in a set of transaction.

Steps:

1. Generate the statistics of the transactions with following fields:

Itemset, Cardinality, Count, Fcount

Where,

Itemset:Id of itemset

Cardinality: is the number of elements in the set.
Count:is the number of occurrence of the item for the respective cardinality.
Fcount: is a cumulative count.

Do the following steps recursively,

2. For generation of k- itemset corresponding cardinality Fcount value is compared with the minsup.
3. If Fcount > = minsup, then retain that item as a candidate item else discard it

4. Perform the join step
5. Perform the pruning step to generate the Candidate itemset (C_K)
6. Checks the itemset against the minsup if it satisfies the condition add it to Frequent itemset (L_K) else discard that itemset.
7. Repeat step 2 to 5 until the no new frequent itemsets are generated, i.e. $L_{K=} \phi$, for some K.

IV. Experimental Results

The experiments are performed on a Intel® Core ™ 2 duo 2.66GHz processor with 4 GB main memory, on Windows 8 platform. All the programs were developed under the java compiler, version 6.

4.1 Data Description

For verifying the usability of our algorithms, we used three of the test datasets made of available to the Workshop on Frequent Itemset Mining Implementations [12].
We use Retail, Mushroom and Accident as real data sets. The test datasets and their properties are described in table 4.1.

Table 4.1 Summaries of characteristics of datasets

Dataset Name	N	|R|	T	L_{max}	Density(%)
Retail	88162	16469	13	76	0.08
Accident	340184	468	34	51	7.26
Mushroom	8124	119	23	23	8.24

N = Number of Transactions in a dataset.
|R| = Number of distinct items.
T = Average transaction length.
L_{max} = Size of the longest transaction.

4.2 Evaluation Metrics

We evaluate our methods to enhance the Apriori algorithm in terms of their **number of candidate itemset generated** before and after pruning operation.
All three programs were run with different set of Minimum support value *(minsup)* and the candidate items accumulated separately before and after pruning action.

4.3 Results

Table 4.2: Number of candidates generated Retail dataset

Retail Dataset	Number of Candidate Sets			
	Static Cardinality Count		Apriori	
Support	Before prune	After Prune	Before prune	After Prune
0.40%	56258	49043	73926	66706
0.50%	24321	20878	42389	38943
0.75%	5940	5071	23286	22412
1%	2218	1926	18822	18517
1.50%	488	386	17033	16926
2%	244	199	16688	16637
3%	134	129	16564	16546

Table 4.3: Number of candidates generated Mushroom dataset

Mushroom Dataset	Number of Candidate Sets			
	Static Cardinality Count		Apriori	
Support	Before prune	After Prune	Before prune	After Prune
60%	103	63	214	174
70%	31	31	145	145
80%	30	24	145	138
90%	11	11	128	126

Table 4.4: Number of candidates generated Retail dataset

Accident Dataset	Number of Candidate Sets			
	Static Cardinality Count		Apriori	
Support	Before prune	After Prune	Before prune	After Prune
90%	31	31	494	494
85.00%	151	71	611	538
80.00%	380	173	838	630
75.00%	1085	360	1539	814

V. Discussion

In order to measure the performance of our methods, we conducted several experiments using the real dataset mentioned in table 4.1. The first one is "Retail', with 16469 items. It consists of 88162 transactions, and average transaction size of 13. It is sparse dataset with few long frequent set. The rest two are 'Accident' and 'Mushroom', both are dense dataset with a lot of long frequent itemsets.

The comparative performance of our proposed method Staitic Cardinality Count, and Apriori Algorithm are shown in table (4.2, 4.3, 4.4).It is been observed that the number of candidate sets generated before and after pruning steps in our proposed method is less as compare to the Apriori Algorithm that leads to save he computational and storage cost.

References

[1]. Agrawal, R. and Srikant, R. 1994. Fast algorithms for mining association rules. In Proc. 20th Int. Conf. Very Large Data Bases, 487-499.

[2]. R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *Proc. of the 1993 ACM SIGMOD Int'l. Conference on Management of Data*, pages 207.216, Washington, D.C., June 1993

[3]. Necip F. Ayan, Abdul Tansel, Erol Arkun, "An Efficient Algorithm To Update Large Itemsets With Early Pruning", KDD-99 San Diego CA USA.

[4]. Charu C. Aggarwal and Philip S. Yu. "Mining large itemsets for association rules", *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 21(1):23-31,March 1998.

[5]. Ming-Syan Chen, Jiawei Han, and Philip S. Yu. "Data mining: An overview from Database Perspective". *IEEE Transactions on Knowledge and Data Engineering.* 8(6):866-883,1996.

[6]. Usama Fayyad. "Mining databases: Towards Algorithms for knowledge discovery". *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 21(1):39-48, March 1998.

[7]. Data Mining : Concepts and Techniques , Jiawei Han, Micheline Kamber ; Morgan Kaufman,200, ISBN 1-55860-489-8

[8]. Han J. and Pei J., "Mining frequent patterns by pattern-growth: methodology and implications". ACM SIGKDD Explorations Newsletter 2, 2, 14-20.

[9]. Heikki Mannila, "Data mining: Machine learning, statistics, and databases", In *Proceedings of 8th Intl. Conf. on Scientific and Statistical Database Management (SSDBM'96),* pages 2-9, Stockholm, Sweden, June 1996.

[10]. J.S. Park, M. Chen, and P.S. Yu, "An Effective Hash Based Algorithm for Mining Association Rules", *Proc. ACM SIGMOD Conf.*, ACM Press, New York, 1995, pp. 175–186.

[11]. S. Brin et al., "Dynamic Itemset Counting and Implication Rules for Market Basket Data" , *Proc. ACM SIGMOD Conf. Management of Data*, ACM Press, New York, 1997, pp. 255–264.

[12]. Workshop on Frequent Itemset Mining Implementations : http://fimi.cs.helsinki.fi/data/