IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 9, Issue 5 (Mar. - Apr. 2013), PP 08-14
www.iosrjournals.org

Comparative Evaluation of Association Rule Mining Algorithms with Frequent Item Sets

Vimal Ghorecha
(Department of MCA, ATMIYA Institute of Technology & Science, India)

Abstract: *This paper represents comparative evaluation of different type of algorithms for association rule mining that works on frequent item sets. Association rule mining between different items in large-scale database is an important data mining problem. Now a day there is lots of algorithms available for association rule mining. To perform comparative study of different algorithms various factor considered in this paper like number of transaction, minimum support and execution time. Comparisons of algorithms are generated based on experimental data which gives final conclusion.*

Keywords – *Apriori, Association Rules, Data Mining, Frequent Pattern*

I. INTRODUCTION

Many change and many of the developments in association rule mining for the last decade are due to new algorithms introduced. On the one hand where the main aim of association rule mining is to provide the better rule and frequent item set to the predict and make decision. This section explains core concept of frequent itemset and association rule mining. Frequent itemset are those items which occur in transaction frequently. For example, a person who purchase Health Policy, generally purchases Accident Policy. This both items frequently appear in database of insurance company.

Association rule mining can be applied in industries like supermarket, insurance company, and online shopping store and has become essential data mining task which gives different rules for taking future decision.

A very popular and firstly introduced algorithm is Apriori algorithm that has been developed in 1993 by Agrawal et al. [1]. After this frequent itemset mining and association rule mining problems have received great deal of attention. Within one or two decade many research paper published and many other algorithms developed based on this Apriori algorithm. An important improvement regarding performance of these algorithms was made and a new technique introduced called frequent pattern tree (FP-tree) [2]. Also related to associated mining algorithm developed called FP-growth.

After this introduction, paper is organized as below. Section 2 gives problem definition. After that each section 3,4,5,6 give brief of different algorithm Apriori, FP-Growth, The Closure Algorithm [3] and The MaxClosure Algorithm [3] respectively. Section 7 & 8 gives comparative evaluation of these algorithms each compared with Apriori and comparison of all algorithms together. Finally paper concluded in section 9.

II. DEFINING PROBLEM

Our problem is generally divided into two different sub-problems. The first part of problem is to find item sets which occurs in database with minimum support; those items sets are known as large item sets or frequent item sets. The second part of problem is to generate association rules from those large item sets with the constraints (also there is other constraints) of minimal confidence.

Let I = { i1, i2, i3, i4……. im } be a set of m different items, D is a database, is a set of database transactions (variable length) where each transaction T contains a set of items i1, i2, i3, i4…….. ik $\subseteq$ I. Each transaction is associated with an identifier, called TID. A formal definition is [4] Let X is a set which contains different items. Any transaction T has element X if and only if it the element X is with that transaction T. An association rule is an implication of the form X $\rightarrow$ Y, where X, Y $\subseteq$ I and X $\cap$ Y = Ø. Here X is called the antecedent and Y is called the consequent of the rule. The rule X $\rightarrow$ Y satisfies in the dataset with confidence c if within all transactions that contain X also contains Y with ratio of c%. The rule X $\rightarrow$ Y has support s in dataset D, if s% of transaction in dataset D either contain X or Y (X $\cup$ Y). The selection of association rules is based on these two values (Other constraints also matters). These are two important measures of rule importunateness. They can be described by the following equations:

$$\text{support}(X \rightarrow Y) = \text{Freq} (X \cup Y) / \text{Total Trns} \quad (1)$$

$$\text{confidence} (X \rightarrow Y) = \text{Freq} (X \cup Y) / \text{Freq}(X) \quad (2)$$

The first rule is {i1, i2, … ,ik-1} $\rightarrow$ {ik}, by checking the confidence. We can create rules from large item sets which satisfies minimum support. First we check that rule can be satisfied by our requirement. If rule is not as per our requirement new rule are generated by removing items from antecedent and inserting this items to consequent. This new rule again checked for confidence and further rule can be satisfied by our requirements.

The above processes continue until the antecedent does not have any item of say it becomes empty. Our second problem is very easy so every researcher concentrates on first problem.

The first sub-problem can be further divided into two sub-problems: candidate large item sets generation process and frequent item sets generation process. We call those item sets whose support exceeds the support threshold as large or frequent item set.

This requires multiple scan of dataset. Each time for checking support and creating rule based on frequent itemset we need to scan whole dataset and we can say all transactions.

We already discuss the process of creating rules in this section which divided into two parts [5]. First one is to identify all frequent item sets. Second generate strong rules which satisfy minimum support and minimum confidence [6].

III. APRIORI ALGORITHM

Apriori is an algorithm [1] is very popular and previous first algorithm for mining frequent item sets. It generates candidate itemset and frequent itemset, like at each level are utilized in next level to generate candidate itemset. As we said earlier it requires multiple database scans, as many as longest as frequent item sets. Apriori employs an iterative approach known as level-wise search, where k item set are used to explore (k+1) item sets. There are two steps in each iteration.

The basic Apriori algorithm is plinth of association rule mining classic Apriori algorithm inspires different researcher how to mine association rules and the market basket analysis example is most popular in 1990s.

L1 = {large 1-itemsets};
for (k = 2; Lk-1 ≠ Ø; k++) do begin
Ck = apriori-gen(Lk-1); //New candidates
forall transactions t ∈ D do begin
Ct = subset(Ck, t);
//Candidates contained in t
forall candidates c ∈ Ct do
c.count++;
end
Lk = { c ∈ Ck | c.count ≥ minsup }
end
Answer = ∪ k Lk;

The first weakness of this algorithm is the generation of a large number of candidate item sets. The second problem is the number of database passes which is equal to the max length of frequent item set.

IV. THE FP-GROWTH ALGORITHM

The FP-growth [2] algorithm for mining frequent patterns with FP-tree by pattern fragment growth is:

A FP-tree constructed with the above mentioned algorithm; where D – transaction database; s – minimum support.

Output of this algorithm is complete set of frequent itemset without generating candidate set and without multiple database scans.

```
Method:
call FP-growth(FP-tree, null).

Procedure FP-growth(Tree, A)
{
        if Tree contains a single path P
        then for each combination (denoted as B) of the nodes in the path P do
                generate pattern B ∪ A with support=minimum support of nodes in B
        else for each ai in the header of the Tree do
        {
                generate pattern B = ai ∪ A with support = ai.support;
                construct B's conditional pattern base and B's conditional FP-tree
                TreeB;
                if TreeB ≠ Ø
                then call FP-growth(TreeB, B)
        }
}
```

V. THE CLOSURE ALGORITHM

```
Candidate = all 1 item sets; KFrequent = empty; Maximal = empty
   while (true) do
      initialize the elements of matrix M with 0;
      count the support for all elements of Candidate
and update their corresponding rows in M;
      for all item sets C in Candidate do
if (C is not frequent)
continue to next itemset;
add C to KFrequent;
compute closure of C in cl(C);
compute all extended closures of C and add them to Maximal;
if no extended closures were found
add cl(C) to Maximal;
      od
      empty Candidate;
      generate new candidate item sets in Candidate based on  KFrequent;
      if Candidate is empty
         break;
      empty KFrequent;
      for all item sets L in Maximal
mark as frequent all item sets from Candidate
      those are contained in L;
      for all item sets C in Candidate
if C is marked as frequent
add C to KFrequent;
      empty Candidate;
      generate new candidate item sets in Candidate based on KFrequent;
      if Candidate is empty
         break;
      empty KFrequent;
     od
   return Maximal
```

Note that the Closure algorithm requires (n/2 + 1) scans of the database where n are the dimension of the longest frequent itemset.

VI. THE MAXCLOSURE ALGORITHM

```
Candidate = all 1-itemsets; Frequent = empty; Maximal = empty;
while (true) do
initialize the elements of matrix M with 0;
count the support for all elements of Candidate
   and update their corresponding rows in M;
for all item sets C in Candidate do
if C is not frequent
   continue to next itemset;
compute closure of C in cl(C);
for all extended closures xcl(C) of C
   if xcl(C) has not been already added to Frequent
      add xcl(C) to Frequent;
if no extended closures were found
   if cl(C) has not been already added to Maximal
      add cl(C) to Maximal;
od
   if Frequent is empty       break;
   empty Candidate;
   move into Candidate all elements from Frequent;
od
return Maximal
```

VII. COMPARATIVE EVALUATION BETWEEN APRIORI, FP-GROWTH, CLOSURE AND MAXCLOSURE ALGORITHM

Four different association rule mining algorithm implemented in Java and tested based on different criteria. The platform specification for this test was:

Intel Core-i3-2330M 2.20GHz processor, with 4 GBRAM, Windows 7 64bit. To study the performance and scalability of the algorithms generated data sets with 10K to 500K transactions, and support factors of 25% and 33% were used. Any transaction may contain more than one frequent itemset. The number of items in a transaction may vary, as well as the dimension of a frequent itemset. Also, the number of items in an itemset is variable. Taking into account these considerations, the generated data sets depend on the number of items in a transaction, number of items in a frequent itemset, etc.

VII.1. Comparative evaluation between Apriori and FP-Growth

From TABLE-1 and Fig.1 we can tell that FP-Growth algorithm is more efficient than classic Apriori algorithm. In analysis process we have taken records from 10K to 500K with support of 33% and 100 numbers of items.

TABLE 1

Records	Apriori (Time in Seconds)	Fp-Growth (Time in Seconds)
10000	1.358	1.352
50000	6.822	6.651
100000	13.273	13.133
300000	40.906	39.657
500000	99.317	89.314

Figure-1

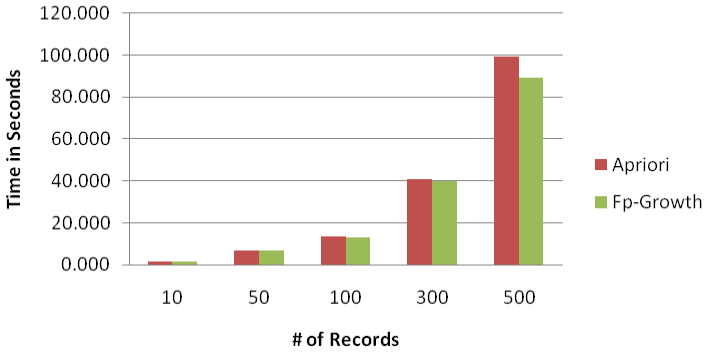


Fig. 1. Apriori vs. Fp-Growth

When we increase the size of record we can see in the table the results are very efficient related to time.

VII.2. Comparative evaluation between Apriori and Closure

From TABLE-2 and Fig.2 we can tell that Closure algorithm is more efficient than classic Apriori algorithm. In analysis process we have taken records from 10K to 500K with support of 33% and 100 numbers of items.

TABLE 2

Records	Apriori (Time in Seconds)	Closure (Time in Seconds)
10000	1.358	0.722
50000	6.822	3.444
100000	13.273	6.671
300000	40.906	20.907
500000	99.317	68.678

Figure-2

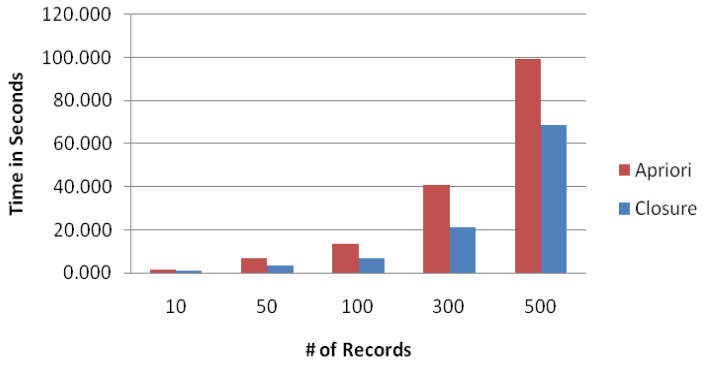


Fig. 2. Apriori vs. Closure

When we increase the size of record we can see in the table the results are very efficient related to time.

VII.3. Comparative evaluation between Apriori and MaxClosure

From TABLE-3 and Fig.3 we can tell that MaxClosure algorithm is more efficient than classic Apriori algorithm. In analysis process we have taken records from 10K to 500K with support of 33% and 100 numbers of items.

TABLE 3

Records	**Apriori (Time in Seconds)**	**MaxClosure (Time in Seconds)**
10000	1.358	0.677
50000	6.822	3.340
100000	13.273	6.563
300000	40.906	20.141
500000	99.317	65.288

Figure-3

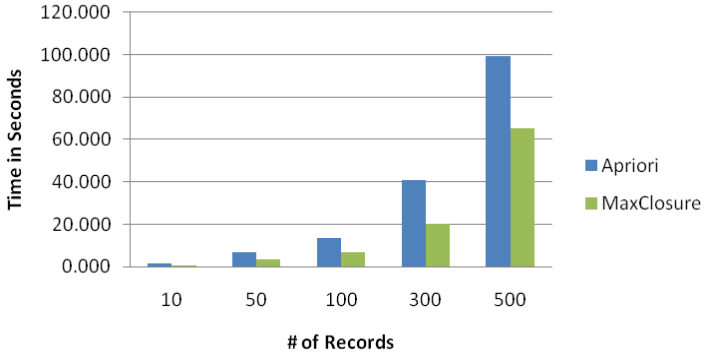


Fig. 3. Apriori vs. MaxClosure

When we increase the size of record we can see in the table the results are very efficient related to time.

VIII. COMPARATIVE EVALUATION BETWEEN ALL ALGORITHMS TOGETHER.

TABLE 4

Records	Apriori (Time in Seconds)	Fp-Growth (Time in Seconds)	Closure (Time in Seconds)	MaxClosure (Time in Seconds)
10000	1.358	1.352	0.722	0.677
50000	6.822	6.651	3.444	3.340
100000	13.273	13.133	6.671	6.563
300000	40.906	39.657	20.907	20.141
500000	99.317	89.314	68.678	65.288

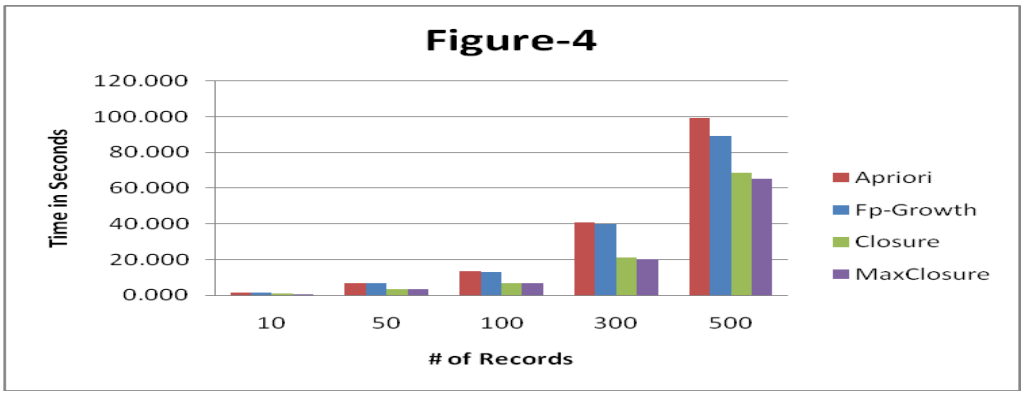
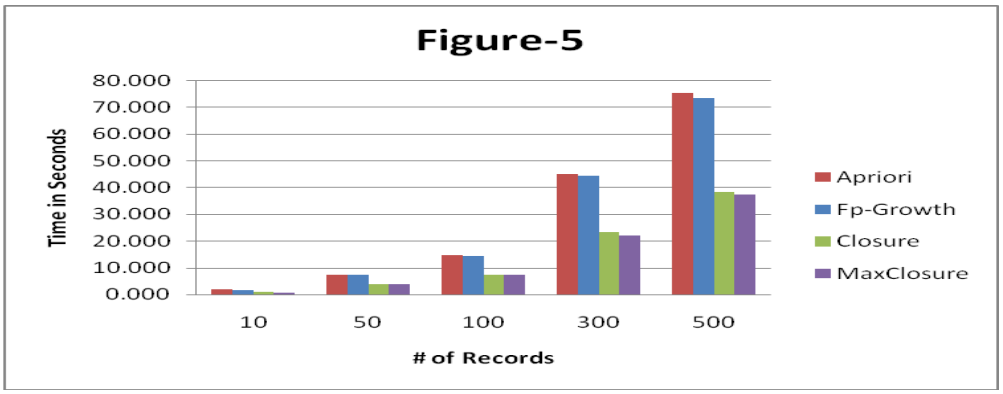


TABLE 5

Records	Apriori (Time in Seconds)	Fp-Growth (Time in Seconds)	Closure (Time in Seconds)	MaxClosure (Time in Seconds)
10000	1.907	1.544	0.803	0.760
50000	7.387	7.411	3.885	3.737
100000	14.822	14.367	7.480	7.270
300000	45.199	44.564	23.461	22.004
500000	75.558	73.640	38.360	37.487



For the comparative study of Classical Apriori, FP-Growth, Closure and MaxClosure Algorithm, we have taken a database of different transaction form 10K to 500K with different number of items 100 and 200.

In this analytical process we considered 500 transactions to generate the frequent pattern with the support count 25% and 33%. We have repeated the same process by increasing the transaction.

Here we first compare the result of all approaches with Classical Apriori Algorithm because all approaches are based on Classical Apriori. After that we compare all approaches to find out the best. After the experiment on all approaches, we have designed a graph and summarized a result in the following table.

IX. CONCLUSION

From above experimental data we can conclude that the MaxClosure algorithm is better than all algorithms. We can also conclude that FP-Growth is better than Apriori, Closure is better than FP-Growth and MaxClosure is better than Closure Algorithm. Here we just consider time as factor. If we consider other factor other than time the result may be vary from factor to factor. Performance of all algorithms is not affected by support factor as well as number of items.

If we closely monitor Table – 5 we can say that time taken by Apriori algorithm when records is 500K and Support is 25%, is less than time taken by Apriori when records is 500K and Support is 33%. So based on data we can conclude that classical Apriori algorithm affected by Support Factor.

If we closely monitor Figure – 5 we can say that time taken by Closure and MaxClosure when records are 500K is same as time taken by Apriori and FP-Growth when records is 300K. So based on data we can conclude that both Closure algorithms take almost half of the time taken by Apriori and FP-Growth.

REFERENCES

[1] R. Agrawal, R. Srikant. "Fast algorithms for mining association rules in large databases" Proc. of 20th Int'l conf. on VLDB: 487-499, 1994.
[2] J. Han, J. Pei, Y. Yin. "Mining Frequent Patterns without Candidate Generation" Proc. of ACM-SIGMOD, 2000.
[3] D. Cristofor, L. Cristofor, and D. Simovici. Galois connection and data mining. *Journal of Universal Computer Science, 6(1)*:60–73, 2000.
[4] R. Győrödi, C. Győrödi. "Architectures of Data Mining Systems". Proc. Of Oradea EMES'02: 141-146, Oradea, Romania, 2002.
[5] M. H. Dunham. "Data Mining. Introductory and Advanced Topics". Prentice Hall, 2003, ISBN 0-13-088892-3.
[6] J. Han, M. Kamber, "Data Mining Concepts and Techniques", Morgan Kaufmann Publishers, San Francisco, USA, 2001, ISBN 1558604898.
[7] R.J. Bayardo, Jr. "Efficiently mining long patterns from databases". In L.M. Haas and A. Tiwary, editors, Proceedings of the 1998 ACM SIGMOD *International Conference on Management of Data, volume 27(2)* of SIGMOD Record, pages 85–93. ACM Press, 1998.
[8] R. J. Bayardo. "Efficiently mining long patterns from databases". In SIGMOD'98, pp. 85-93.
[9] G. Dong and J. Li. "Efficient mining of emerging patterns: Discovering trends and differences". In KDD'99, pp. 43-52.
[10] Zaki, M.J. and Hsiao, C.J. "CHARM: An efficient algorithm for closed itemset mining". In Proc. SIAM Int. Conf. Data Mining, Arlington, VA, pp. 457–473, 2002.