Comparative Study of Association Rules Mining

Thi Dar Aye, Khin Mar Soe
University of Computer Studies, Yangon, Myanmar
thidaraye925@gmail.com, kmsucsy@gmail.com

Abstract

Association rule mining is a technique to find useful patterns and associations in transactional databases. The mining of association rules can be mapped into the problem of discovering large (frequent) itemsets where is a grouped of items which appear in a sufficient number of transaction. The discovery of interesting association relationships among huge amount of business transaction records can help in many business decision making process . There are many association rules mining algorithms. But this system is intended to make the comparative study of three association rules mining algorithms such as DHP algorithm, PHP algorithm and Hybrid Approach of Support-Ordered Tree and PHP based on same dataset. Both DHP and PHP algorithm use hash base method and pruning method to reduce database size. DHP use direct hashing technique. PHP use perfect hashing technique. The two dataset, Kyar Nyo Pan Stationary Store and Orange minimarket are used.

Keywords: association rule, database, frequent pattern , itemset.

1. Introduction

Data mining has attracted a growing amount of attention in database communities due to its wide applicability in retail industry for improving marketing strategy. Since the amount of sales data is huge, it is important to implement efficient algorithms to conduct mining on these data.

One of the most important data mining problems is mining association rules. Mining association rules can be decomposed into two sub-problems. First , it is needed to identify all sets of items (itemsets) that are contained in a sufficient number of transactions above the minimum support (requirement). These itemsets are referred to as large itemsets. Once all large itemsets are obtained, the desired association rules can be generated in a straightforward manner. The overall performance of association rules mining is depended on the first step. Therefore, many researcher focused on the performance of the finding frequent itemsets. This system compare algorithm DHP, PHP, Hybrid approach of Support-Order Tree and PHP for efficient large itemset generation.

2. Association Rules Mining

Mining association rules are to find interesting association or correlation relationships among a large set of data that frequently occur together, and then formulate rules that characterize these relationships. Mining association rules is composed of the following two steps –

- To discover the large itemsets, i.e., all sets of large itemsets.
- To use the large itemsets to generate the association rules for the database.
- The overall performance of mining association rules is determined by the first step. Let I= {i1,i2,…,im} be a set of literals, called items. Let D be a set of transactions, where each transaction, T is a set of items such that T $\subseteq$ I. Each transaction is associated with an identifier, called TID. Let X be a set of items. A transaction T is said to contain X if and only if X $\subseteq$ T. An association rule is an implication of the form X $\Rightarrow$Y, where X$\subset$ I , Y $\subset$ I and X $\cap$ Y=ф. The rule X $\Rightarrow$ Y holds in the transaction set D with confidence c if c% of transactions in D that contain X also contain Y. The rule X $\Rightarrow$ Y has support s in the transaction set D if s% of transactions in D contains X $\cup$ Y. [1]

3. Direct Hashing and Pruning (DHP) Algorithm

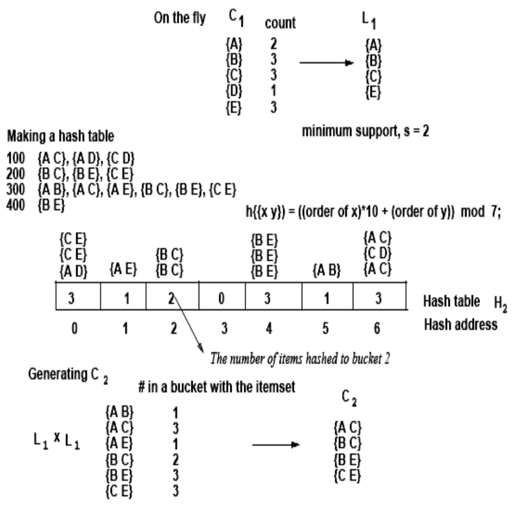
DHP algorithm is based on Apriori algorithm. This algorithm will utilize a hash method for candidate itemset generation during the initial iterations and employ pruning techniques to progressively reduce the transaction database size. It has two major features:

- one is efficient generation of large itemsets
- other is effective reduction on transaction database size.

As in Apriori, in each pass the set of large itemsets, Li, is used to form the set of candidate large itemsets Ci+1 by joining Li, with Li on (i-1). Then the database is scanned and it will count the support of each itemset r the processing cost of determining Li will be. By constructing a significantly smaller C2, DHP can also generate a much smaller D3 to derive C3.[2]

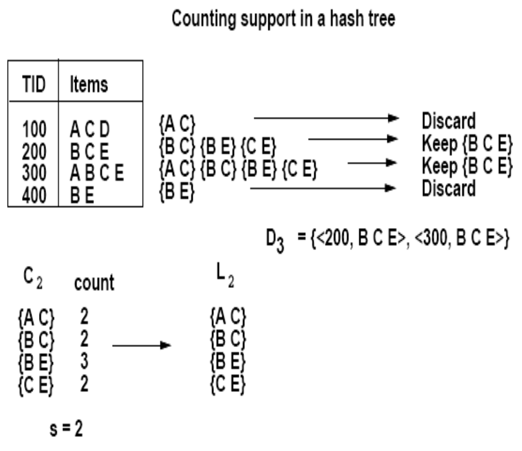
During the first pass of DHP algorithm, get a set of large 1-itemsets and makes a hash tables (H2) for 2-itmsets. In later passes, generates the set of candidate itemsets Ck based on the hash table Hk generated in previous pass, determines the set of large k-itemsets Lk, reduces the size of databases for the next large itemsets and makes a hash table for candidate large (k+1)-itemsets.The same process is perform until reduce database is empty.

DHP reduces the database size progressively by not only trimming each individual transaction size but also pruning the number of transactions in the database. Any subset of a large itemset must be a large itemset by itself. This fact suggests that a transaction be used to determine the set of large (k+1)-itemsets only if it consists of (k+1) large k-itemsets in the previous pass.



On the fly C_1 count L_1
{A} 2, {B} 3, {C} 3, {D} 1, {E} 3
{A} {B} {C} {E}
Making a hash table
minimum support, s = 2
100 {A C}, {A D}, {C D}
200 {B C}, {B E}, {C E}
300 {A B}, {A C}, {A E}, {B C}, {B E}, {C E}
400 {B E}
h({x y}) = ((order of x)*10 + (order of y)) mod 7;
Hash table H_2
Hash address
The number of items hashed to bucket 2
Generating C_2
# in a bucket with the itemset
$L_1 \times L_1$
C_2

Figure 1. Sample Generation of candidate itemsets with DHP Algorithm



Counting support in a hash tree
TID Items
100 A C D
200 B C E
300 A B C E
400 B E
{A C} Discard
{B C} {B E} {C E} Keep {B C E}
{A C} {B C} {B E} {C E} Keep {B C E}
{B E} Discard
D_3 = {<200, B C E>, <300, B C E>}
C_2 count L_2
{A C} 2, {B C} 2, {B E} 3, {C E} 2
{A C} {B C} {B E} {C E}
s = 2

Figure 2. Sample Reducing Database size

4. Perfect Hashing and Database Pruning (PHP) Algorithm

The PHP algorithm is based on DHP algorithm. The difference of PHP algorithm is that, it uses perfect hashing in order to create a hash table for the candidate *k*+1 itemsets. As perfect hashing is used, the hash table contains the actual counts of the candidate *k*+1 itemsets. Hence we do not need to make extra processing to count the occurrences of candidate *k*+1 itemsets as in the DHP algorithm. The algorithm also prunes the database at each step in order to reduce the search space.

So we call the algorithm as Perfect Hashing and Pruning (PHP) and the algorithm is as follows: During the first pass of PHP algorithm, a hash table with size equal to the distinct items in the database is created. Each distinct item in the database is mapped to different location in the hash table, and this method is called as *perfect hashing*. The hash table adds a new entry if an entry for item *x* does not exist in the hash table and initializes its count to 1, otherwise it increments the count of *x* in the table by 1. After the first pass, the hash table contains the exact number of occurrences of each item in the database. By only making one pass over the hash table, which is in memory, the algorithm easily generates the frequent 1-itemsets. After that operation, the hash table prunes all the entries whose support is less than the minimum support. In the subsequent passes, the algorithm prunes the database by discarding the transactions, which have no items from frequent itemsets, and also trims the items that are not frequent from the transactions. At the same time, it generates candidate *k*-itemsets and counts the occurrences of *k*-itemsets. At the end of the pass, *Dk* contains the pruned database, *Hk* contains the occurrences of candidate *k*-itemsets, and *Fk* is the set of frequent *k*-itemsets. This process continues until no new *Fk* is found.[3]

TID	Items
100	A C D
200	B C E
300	A B C E
400	B E

A	C	D	B	E
2	3	1	3	3

L1	count
{A}	2
{B}	3
{C}	3
{E}	3

Making a hash table H2
100 {AC},{AD},{CD}
200 {BC},{BE},{CE}
300 {AB},{AC},{AE},{BC},{BE},{CE}
400 {BE}

{AC}	{BC}	{BE}	{CE}	{AB}	{AE}
2	2	3	2	1	1

L2	count
{AC}	2
{BC}	2
{BE}	3
{CE}	2

Figure 3. Sample Generation of frequent itemsets with PHP Algorithm

5. Supported-Order Tree

The Supported-Order Tree is constructed by extracting 1-itemsets and 2-itemsets from all transaction and using them to update the Support-Order Tree. The Support–Order Tree is ordered by support count. It has two levels of nodes (excluding the special ROOT node). The bracket number beside a node's label denotes the support count. The Support-Order Tree is constructed without the need to know the support thresholds; it is support-independent. Its main advantage lies in its speed in discovering L1 and L2. L1 and L2 can be found promptly because there is no need to scan the database. In addition, the search (depth-first) can be stopped at a particular level the moment a node representing a nonfrequent itemsets is found because the nodes are all support ordered. Its main weakness is that it can only discover L1 and L2. Thus later passes are discovered by PHP algorithm.

A set of Support-Order Tree is built from a database to store support counts of all 1-itemsets and 2-itemsets. We use a special node called ROOT to link all these Support-Order Tree together and keep them ordered by support count in a way similar to their second-level nodes.[4]

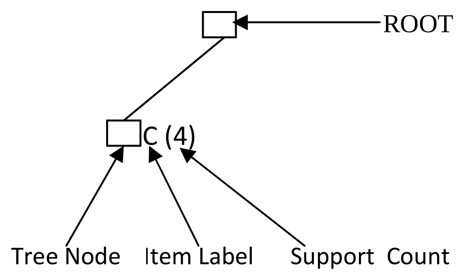


Figure 4. Sample of Support-Order Tree

TID	Items
100	A C D
200	B C
300	A B C
400	C D

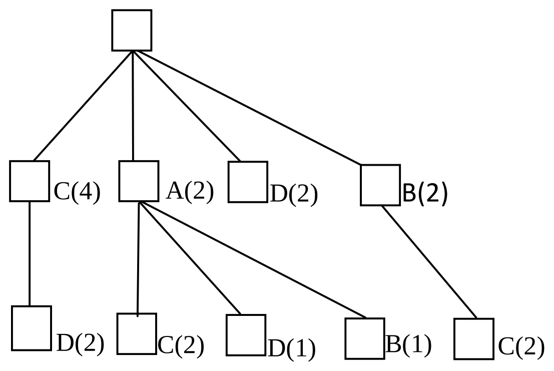


Figure 5. Sample Generation of frequent itemset with Support-Order Tree

6. Implementation of the System

In this system, user choose algorithm and then input a minimum support to the system. The system runs the algorithm by using the minimum support and the dataset. And then return the result to the user. After finding all of the result of three frequent itemsets algorithm (DHP, PHP, Hybrid), generate the comparison chart.

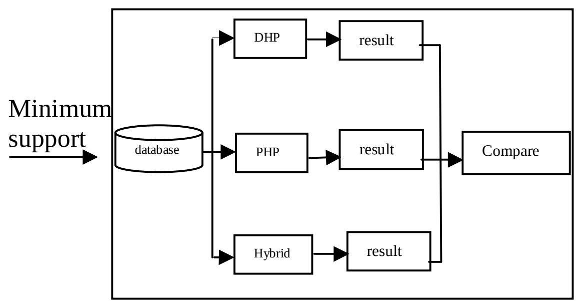


Figure 6. System Design of our proposed system

7. Experimental Results

All of three algorithm are tested on two different datasets. There are Kyar Nyo Pan Stationary dataset with average transaction length 9 and the number of transaction 5000 and Orange minimarket dataset with average transaction length 6 and the number of transaction 5000. The frequent itemsets found by the three algorithms are the same. Figure 7 and Figure8 shows that Hybrid Approach of Support-Order tree algorithm , for all minimum support values, performs better than other two algorithm. Since L1 and L2 can be found quickly because there is no need to scan the database. PHP algorithm performs better than DHP algorithm for all minimum support values. Since PHP does not perform extra processing for counting the occurrences of each itemset as in DHP.

Figure 7. Support and Run time (sec) by using Kyar Nyo Pan Stationary dataset

Figure 7. represents number of transaction is 5000, Minimum support thresholds are 5%, 10%, 15%, 20%. This result between the minimum support threshold and runtime (seconds).

Figure 8. Support and Run time (sec) by using Orange minimarket dataset

Figure 8. represents number of transaction is 5000, Minimum support thresholds are 5%, 10%, 15%, 20%. This result between the minimum support threshold and runtime (seconds).

8. Conclusion

Data mining in large database is one of today's real challenges to database research area. Since, the amount of sale data is huge, it is important to implement efficient algorithm to conduct mining on these data. A fundamental component in data mining tasks is finding frequent itemsets in a given dataset. In this system, the performance of the Hybrid Approach of Support-Order Tree and PHP algorithm show that it is efficient for mining both long and short frequent patterns and is faster than the DHP and PHP Algorithm.

9. References

[1].Jiawei Han and Micheline Kamber. Data Mining :Concepts and Techniques. Morgan Kaufmann,2000.

[2].J. S. Park, M. S. Chen and P. S. Yu, "An Effective Hash-Based Algorithm for Mining Association Rules. IBM Thomas J Watson Research Center.

[3]. S.Ayse Ozel and H. Altay Giivenire, "An Algorithm for Mining Association Rules Using Perfect Hashing and Database Pruning", Bilkent University, Department of Computer Engineering, Ankara,Turkey.{selma,guvenir}@cs.bilkent.edu.tr

[4].Yew-Kwong Woon,Wee-Keong Ng,Member, "A Support-Ordered Trie for Fast Frequent ItemSet Discovery" , IEEE Computer Society, and Ee-Peng Lim, Senior Member , IEEE,Vol.16,No.7,July 2004.