ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 10, April 2014

An Efficient Algorithm for Mining Association Rules for Large Itemsets in Large Databases

Dr.Hussam Al-Shorman, Dr.Yosef Hasan Jbara

Abstract*— The data mining of association rules between items in a large database is an essential research aspect in the data mining fields. Discovering these associations is beneficial to the correct and appropriate decision made by decision-makers. A well-known problem that limits the practical usage of association rule mining algorithms is the extremely large number of rules generated. Such a large number of rules makes the algorithms inefficient and makes it difficult for the end users to comprehend the discovered rules. In this paper we present an efficient algorithm for mining association rules in large transactional databases. This algorithm is referred to as the CGDBT – compact grouping database transactions. It uses an efficient approach to convert the original transactional database to compact transaction groups for efficient mining. This allows generating frequent patterns quickly by skipping the repetitive database scan and reducing a great amount of time per database scan. We have performed extensive experiments and compared the performance of our algorithm with one of the best existing algorithms. Experimental results show that our approach outperforms that algorithm and show that our approach is effective, efficient and promising.

***Index Terms*—Mining Association Rules, Large Itemsets.**

I. INTRODUCTION

Number of data mining algorithms have been recently developed that greatly facilitate the processing and interpreting of large stores of data. One example is the association rule mining algorithm, which discovers correlations between items in transactional databases. An association rules mining is motivated by decision support problems faced by most business organizations and is described as an important area of research. One of the main challenges in association rules mining is developing fast and efficient algorithms that can handle large volumes of data because most mining algorithms perform computation over the entire database and often the databases are very large. Mining association rules may require iterative scanning of large databases, which is costly in processing.

Many algorithms have been discussed in the literature for discovering, association rules ([1], [2], [3], [4], [5], [6], [7]). A milestone in these studies is the development of an Apriori-based, level-wise mining method for associations, which has sparked the development of various kinds of Apriori-like association mining algorithms. There is an important, common ground among all these methods developed: the use of an Apriori property of frequent patterns: if any length-k pattern not frequent in the database, none of its length-(k + 1) super-patterns can be frequent. This property leads to the powerful pruning of the set of itemsets to be examined in the search for longer frequent patterns based on the existing ones. Besides applying the Apriori property, most of the developed methods adopt a level-wise, candidate generation-and test approach, which scans the database multiple times (although there have been many techniques developed for reducing the number of database scans). The first scan finds all of the length-1 frequent patterns. The k-th (for k >1) scan starts with a seed set of length (k -1) frequent patterns found in the previous pass and generates new potential length-k patterns, called candidate patterns. The k-th scan of the database finds the support of every length k candidate pattern. The candidates which pass the minimum support threshold are identified as frequent patterns and become the seed set for the next pass. The computation terminates when there is no frequent pattern found or there is no candidate pattern that can be generated in any pass. The candidate generation approach achieves good performance by reducing the number of candidates to be generated. However, when the minimum support threshold low or the length of the patterns to be generated is long, the candidate generation-based algorithm may still bear the following non-trivial costs, independent of detailed implementation techniques.

1. The number of candidates to be generated may still be huge, especially when the length of the patterns to be generated is long. For example, to generate one frequent pattern of length 100, such as {al, a2 . . . . , al00}, the number of candidates that has to be generated will be at least

$$\Sigma_{i=1}^{100}\binom{100}{i} = 2^{100} - 1 \approx 10^{30}.$$

2. Each scan of the database examines the entire database against the whole set of Current candidates, which is quite costly when the database is large and the number of candidates to be examined are numerous.

To overcome this difficulty, a new approach, called CGDBT: Compact Grouping Database Transactions, has been developed, which is fundamentally different from all the previous algorithms in that it adopts grouping database transactions philosophy to obtain non-redundant transactions. The method preserves the essential groupings of the original data elements for mining. Then the analysis is focused on counting the frequency of the relevant data sets. Instead of scanning the entire database to match against the whole corresponding set of candidates in each pass, the method scans the obtained group data sets and takes into account the frequency of each data set. Such methodology substantially reduces the search space and leads to high performance.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 10, April 2014

We have performed extensive experiments and compared our algorithm with one of the best previous algorithms. Our experimental study shows that for computationally intensive cases, our algorithm performs better than the previous algorithm in terms of CPU time.

The paper is organized as follows: in the next section, we give brief description of the association rules mining. In Section 3, we give an overview of the previous algorithms. In section 4, we describe our algorithm. Performance results are described in section 5. Section 6 contains conclusion and future work.

II. ASSOCIATION RULES MINING

The mining association rules basically are to find important associations among items in a given database of sales transactions such that the presence of some items will imply the presence of other items in the same transaction. A formal model is introduced in [9]. Let I = {i1, i2, …, im} be a set of binary attributes, called items. Let D a set of transactions and each transaction T is a set of items such that T⊆I. Let X be a set of items. A transaction T is said to contain X if and only if X⊆T. An association rule is an implication of the form X⇒Y, where X⊂ I, Y⊂ I, and X∩Y = ∅. Furthermore, the rule X⇒Y is said to hold in the transaction set D with confidence c if there are c% of the transaction set D containing X also containing Y. The rule X⇒Y is said to have support s in the transaction set D if there are s% of transactions in D containing X∪Y. The confidence factor indicates the strength of the implication rules; whereas the support factor indicates the frequencies of the occurring patterns in the rule. The goal of mining association rules is then to find "strong" rules, which are rules with high confidence and strong support, in large databases.

It has been shown that the problem of discovering association rules can be reduced to two sub-steps:

1) Find all frequent itemsets for a predetermined support

2) Generate the association rules from the frequent itemsets.

III. PREVIOUS WORK

Since its introduction in 1993 [1], many algorithms with different approaches have been suggested for the ARM problem. Here we review some of the related work that forms a basis for our algorithm.

IV. ASSOCIATION RULES MINING

A. The Apriori Algorithm

The Apriori algorithm first counts occurrences of items to determine the large 1-itemsets. (Itemset with the cardinality of 1) Then there are two phases in the subsequent passes. First of all, the large itemsets Lk-1 found in the (k-1) th pass are used to generate the candidate itemsets Ck. Next, the support of candidates in Ck is counted by scanning the database. The final answer is obtained by taking the union of all Lk itemsets. The function takes Lk-1 as argument and returns a superset of the set of all large k-itemsets. Firstly, the join step is taken by joining Lk-1 with Lk-1. The next step is the prune step, which deletes all itemsets c∈Ck such that some (k-1)-subset of c is not in Lk-1.

The AprioriTid algorithm is a variation of the Apriori algorithm. The AprioriTid algorithm also determines the candidate itemsets before the pass begins. The main difference from the Apriori algorithm is that the AprioriTid algorithm does not use the database for counting support after the first pass. Instead, the set <TID, {Xk}> is used for counting. (Each Xk is a potentially large k-itemset in the transaction with identifier TID.) The benefit of using this scheme for counting support is that at each pass other than the first pass, the scanning of the entire database is avoided.

B. DHP Algorithm

The DHP (Direct Hashing and Pruning) algorithm is an effective hash-based algorithm for the candidate set generation. The DHP algorithm consists of three steps. The first step is to get a set of large 1-itemsets and constructs a hash table for 2-itemsets. (The hash table contains the information regarding the support of each itemset.) The second step generates the set of candidate itemsets Ck, but it only adds the k-itemset into Ck if that k-itemset is hashed into a hash entry whose value is greater than or equal to the minimum transaction support. The third part is essentially the same as the second part except it does not use the hash table in determining whether to include a particular itemset into the candidate itemsets.

C. Partition Algorithm

The Partition algorithm logically partitions the database D into n partitions, and only reads the entire database at most two times to generate the association rules. The reason for using the partition scheme is that any potential large itemset would appear as a large itemset in at least one of the partitions.

D. FP-Growth Algorithm

The Apriori algorithm has its own disadvantages that made other scholars think of new approaches to mine frequent patterns. The 2 main downsides are the possible need of generating a huge number of candidates if the number of frequent 1-itemsets is high or if the size of the frequent pattern is big the database has to be scanned repeatedly to match the candidates and determine the support. The frequent-pattern growth (FP-growth) algorithm adopts a divide-and-conquer strategy and a frequent-pattern tree to mine the frequent patterns without candidate generation which is a big improvement over Apriori.

V. GDBT FOR MINING FREQUENT PATTERNS

The proposed method executes in two phases: Database Preprocessing and discovering the association rules. The first phase consists of two steps. In the first step, the CGDBT

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 10, April 2014

algorithm logically sorts the database itemsets in ascending order. Then the algorithm performs an item-based counting to construct a compact table of items along with a frequency of each item. At the end of step I, each entry in the generated compact table represents a group of database itemsets. To count the number of occurrences for each candidate itemset Ck, the proposed method does not scan the whole database. Instead, only the compact table which represents the database transactions is accessed, and the count is obtained directly without the need to scan the whole database transactions. In other words, in step 1, GDBT performs item-based database grouping when the database is large then switches to main-memory-based mining by constructing a compact table and transforming mining database into mining this compact table. In the second step, the missing values are manipulated. Each missing value is replaced by a probability distribution calculated using the existing data. This probability distribution represents the likelihood of possible values for the missing data, calculated using frequency counts from the entries that do contain data for the corresponding field. After preprocessing is done on the database itemsets, resulting data should be converted into the format that can be inputted for generating rules.

The second phase is composed of 2 steps. First, all of the transactions are read from a database. Second, the actual supports for these itemsets are generated and the large itemsets that meet the support and confidence thresholds are identified. If the database entry completely matches the corresponding item, the support is incremented by the absolute value of the difference between the values, divided by the maximum possible value for the given item. To efficiently generate the candidate itemsets, we store the itemsets in sorted order. We also store references to the itemsets in a hash table for performing pruning efficiently. Finally, discovering rules. Once the large itemsets and their supports are determined, the rules can be discovered in a straight forward manner as follows: if I is a large itemset, then for every subset a of I, the ratio support (l) / support (a) is computed. If the ratio is at least equal to the user specified minimum confidence, then the rule a $\Rightarrow$ (1- a) is output. Multiple iterations of the discovery algorithm are executed until at least N itemsets are discovered with the user specified minimum confidence, or until the user-specified minimum support level is reached.

The algorithm uses Leverage measure introduced by Piatetsky (11) to filter the found item sets and to determine the interestingness of the rule.

$$leverage(X \rightarrow Y) = P(X \quad and \quad Y) - P(X)P(Y)$$

Leverage measures the difference of X and Y appearing together in the data set and what would be expected if X and Y where statistically dependent. Using minimum leverage thresholds at the same time incorporates an implicit frequency constraint. e.g., for setting a min. leverage thresholds to 0.01% (corresponds to 10 occurrence in a data set with 100,000 transactions) one first can use an algorithm to find all itemsets with minimum support of 0.01% and then filter the found item sets using the leverage constraint. By using Leverage measure we reduce the generation of candidate's itemsets and thus we reduce the memory requirements to store a huge number of useless candidates.

VI. EXPERIMENTAL RESULTS

To evaluate the efficiency of the proposed algorithm, we have extensively studied our method performance by comparing it with the FP-tree algorithm [10]. The proposed algorithm is implemented using Microsoft Visual Basic for applications (VBA) on a Pentium IV 3.7 GHz PC with 2 GB of main memory. The database is stored on an attached 40GB disk.

The test database is the transaction database provided with Microsoft SQL Server 2000. Three data sets of 24,000, 45,000, and 60,000 transaction records of experimental data are randomly sampled from the transaction database. The data sets contain 210, 230, and 365 items respectively; with the longest transaction record contains 12, 15, 21 items respectively. The experiment results of applying our algorithm are compared to those for applying the standard FP-tree algorithm under the various minimum supports threshold, which are set at 0.58%, 0.52%, 0.48%, and 0.44%. We have observed considerable reduction in the number of association rules generated by our algorithm, as compared to the standard FP-tree algorithm. This reduction is also dependent on the support and confident values used; e.g., for data set 1, we observed a reduction when the values support = 58.0% and confidence = 60.0% were used. For data set 2, we observed a reduction when the values support = 52.0% and confidence = 60.0% were used, and for data set 3; when the values support = 52.0% and confidence = 60.0% were used. We present below in table 1.

The Table shows that our algorithm outperforms FP-growth on some support thresholds and number of frequent itemsets that created is less than the other that obtains from the FP-Tree.

TABLE I

	Data set 1	Data set 2	Data set 3
No. of transactions	24,000	45,000	60,000
No. of items	210	230	365
Max items / transaction	12	15	20
Min items / transaction	1	1	4
Support %	**0.58%**, 0.52%, 0.48%, 0.44%	0.58%, **0.52%**, 0.48%, 0.44%	**0.58**%, 0.52%, 0.48%, 0.44%
Confidence %	**0.60%**, 0.55 %, 0.50 %	**0.60**%, 0.55 %, 0.50 %	0.60%, **0.55** %, 0.50 %
No. of rules / FP-tree	**4856**	**6058**	**7482**
No. of rules / GDBT	**4438**	**5597**	**7011**

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 3, Issue 10, April 2014

VII. CONCLUSION

In this paper we presented fast and scalable algorithm which is not only efficient but also fast for discovering association rules in large databases. We compared our algorithm to the previously known algorithm, the FP-growth algorithm. We presented experimental results, showing that the proposed algorithm outperform FP-growth. An important, contribution of our approach is that it drastically reduces the CPU time associated with previous algorithm. We have demonstrated with extensive experiments that the number of rules generated is actually less than the best existing algorithm for some supports thresholds. We demonstrate the effectiveness of our algorithm using sample databases. We develop a visualization module to help the user manage and understand the association rules. In short, proposed algorithm is efficient and highly scalable for mining very large databases.

Future work includes applying these algorithms to real data like retail sales transaction, medical transactions, WWW server logs, etc. to confirm the experimental results in the real life domain, since the proposed algorithm is evaluated only with test cases.

ACKNOWLEDGMENT

This work was made possible by a grant from Al-balqa Applied University – Jordan.

REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules between Sets of Items in Large Databases," Proc. ACM SIGMOD, May 1993.

[2] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules in Large Databases," Proc. 20th Int'l Conf. of Very Large Data Bases, Sept. 1994.

[3] J.-S Park, M.-S. Chen, and P.S. Yu, "An Effective Hash Based Algorithm for Mining Association Rules," Proc. ACM SIGMOD, May 1995.

[4] J. Han, J. Pei, Y. Yin. "Mining Frequent Patterns without Candidate Generation". Proc. of ACM-SIGMOD, 2000.

[5] R.C. Agarwal, C.C. Aggarwal, and V.V.V. Prasad. Depth first generation of long patterns. In Ramakrishnan et al. [32], pages 108–118.

[6] R.C. Agarwal, C.C. Aggarwal, and V.V.V. Prasad. A tree projection algorithm for generation of frequent itemsets. Journal of Parallel and Distributed Computing, 61(3):350–371, March 2001.

[7] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A.I. Verkamo. Fast discovery of association rules. In U.M. Fayyad, G. Piatetsky- Shapiro, P. Smyth, and R. editors, Advances in Knowledge Discovery and Data Mining, pages 307–328. MIT Press, 1996.

[8] Generating Non-Redundant Association Rules. Mohammed J. Zaki Computer Science Department, Rensselaer Polytechnic Institute, Troy NY 12180.

[9] R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules between Sets of Items in Large Databases," Proc. ACM SIGMOD, May 1993.

[10] J.Han, J.Pei, Y.Yin, and R.Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach [J]. Data Mining and Knowledge Discovery, 2003.8(1):53-87.

[11] Piatetsky-Shapiro, G., Discovery, analysis, and presentation of strong rules. Knowledge Discovery in Databases, 1991: p. 229-248.

AUTHOR BIOGRAPHY

Hussam Al-Shorman is a lecturer in Al-balqa Applied University – Jordan. He received a Ph.D. in Artificial Intelligence from the Faculty of Computer Information Systems, University of Banking and Financial Sciences. His research interest in neural networks, artificial intelligence and software engineering areas. He has a wealth of expertise gained from his work experiences in Jordan, ranging from systems programming to network administration. His email address is < Shormanhussam@yahoo.com.>.

Yosef Jbara is a lecturer in the Department of Computer Sciences at Buraydah Private Colleges – Al Qaseem- Kindom of Saudi Arabia. He received a Ph.D. in Artificial Intelligence from the Faculty of Computer Information Systems, University of Banking and Financial Sciences. He has published in the areas of artificial intelligence; simulation modeling; data mining and software engineering. His research focuses on artificial intelligence areas as well as simulation and data mining. He has a wealth of expertise gained from his work experiences in Jordan and Saudi Arabia, ranging from systems programming to computer network design and administration. His email address is <yosefjbara@yahoo.com.>.