Comparative Study of Apriori-variant Algorithms

Sofianita Mutalib, Ammar Azri Abdul Subar, Shuzlina Abdul-Rahman, and Azlinah Mohamed

Universiti Teknologi MARA (Shah Alam), Malaysia, {sofi, shuzlina, azlinah}@tmsk.uitm.edu.my

ABSTRACT

Big Data era is currently generating tremendous amount of data in various fields such as finance, social media, transportation and medicine. Handling and processing this "big data" demand powerful data mining methods and analysis tools that can turn data into useful knowledge. One of data mining methods is frequent itemset mining that has been implemented in real world applications, such as identifying buying patterns in grocery and online customers' behavior. Apriori is a classical algorithm in frequent itemset mining, that able to discover large number or itemset with a certain threshold value. However, the algorithm suffers from scanning time problem while generating candidates of frequent itemsets. This study presents a comparative study between several Apriori-variant algorithms and examines their scanning time. We performed experiments using several sets of different transactional data. The result shows that the improved Apriori algorithm manage to produce itemsets faster than the original Apriori algorithm.

Keywords: Apriori, Association Rule Mining, Frequent Itemset Mining

I INTRODUCTION

Frequent itemset mining is one of the popular techniques in discovering interesting associations among items in database. For example, the resulted associations could be useful information for marketing and determining good prices for products based on customers' needs. Frequent itemset mining has been used widely in recommendation systems such as in market basket analysis in hypermarket. Recommendation systems are widely used to predict what users are looking for on various kind of things, such as books, movies, music and so on. In recent years, recommendations can be generated from algorithms, such as Apriori. The implementation of recommendation and suggestion systems can be seen in a lot of search tabs, such as in YouTube and Amazon. Recommendation systems can reduce searching time while clicking and scrolling the pages.

Apriori is a multi-pass algorithm; where candidate of itemsets are formed while passing the database by extending prior frequent itemsets with each transaction items. However, lot of candidates of itemset may be infrequent and the process of passing the database is very time consuming. Apriori applied downward closure property, which refers to an itemset is frequent only if all its subsets are frequent. This means that if {diaper} was found to be infrequent, we can expect {diaper, pizza} to be equally or even more infrequent. So in consolidating the list of popular itemsets, we need not consider {diaper, pizza}, nor any other itemset configuration that contains diaper.

Generally, Apriori uses a "bottom up" approach, where the algorithm starts by finding frequent one itemset and extending one item at a time through candidate generation process. It generates candidate itemsets of length k from item sets of length k-1. Then it prunes the candidates which have an infrequent sub pattern. Next, the groups of candidates are tested against the database. It scans the transaction database to determine frequent itemsets among the candidates. The algorithm terminates when no further successful extensions are found. Apriori uses breadth-first search to count candidate item sets efficiently.

In this paper, we prepared an experimental study using Apriori based algorithms in mining itemsets. The rest of the paper is organized as follows. The description of related studies in frequent itemset mining using Apriori based algorithms is given in Section II. Section III describes the experimental setup and Section IV describes the results. Finally, the conclusions are presented in Section V.

II FREQUENT ITEMSET MINING USING APRIORI

The inspirations for association rule mining originally came from market basket analysis. A market basket basically consists of a collection of items purchased by a customer in one transaction. If we investigate the customers' transactions, we will able to discover group of items that were highly purchased by customers. For example, we found a rule that if customer *A* buys milk then customer *A* buys coffee also. So, from this rule, there is a high chance that customer *B* who buys milk, will also buy coffee. Association rule mining can be generalized to the analysis of sequences, which is called as sequence mining.

The entire dataset, as shown in Table 1, is a sample of transactional data. Association rule mining includes two main processes:

- finding all frequent itemsets with certain support value in the transactional data.

- generating strong association rules from the frequent itemsets that meet confidence threshold.

From both procesess, the itemsets and the set of rules will be discovered and can be evaluated as useful knowledge to the domain. Next, we reviewed Apriori algorithm for association rule mining.

Table 1. Sample of Transactional Data, adapted from (Han & Kamber, 2006)

Transaction	Item
1	i_1, i_2, i_5
2	i_2, i_4
3	i_2, i_3
4	i_1, i_2, i_4
5	i_1, i_3
6	i_2, i_3
7	i_1, i_2, i_3, i_4

A. Apriori Algorithm

Apriori algorithm was proposed in 1993 by Agarwal (1994). This algorithm is widely used because it is very simple and easy to be implemented in mining all frequent itemsets in database. The algorithm is basically generating candidate itemsets of a given size, *k*-itemsets, then scan the database to check, and counts the number of occurrence of each item in the database. Figure 1 shows the pseudo code for Apriori algorithm. The pseudo code for the algorithm is given below for a transaction database *T*, and a support threshold of *minsup*. C_k is the candidate set for level *k*.

Algorithm 1 *a priori* algorithm

$C_1 := \{\{i\} : i \in J\}$ {set of all 1-itemsets}, $k = 1$
while $C_k \neq \emptyset$ **do**
  For all $A \in C_k$ determine $s(A)$ by scanning database.
  $L_k :=$ set of all frequent itemsets in C_k.
  Construct join $L_k * L_k$.
  Prune result by removing sets with infrequent subsets to get C_{k+1}.
  $k := k + 1$
end while
return all frequent itemsets $\bigcup L_k$

Figure 1. Apriori Algorithm (Han & Kamber, 2006)

At each step, the algorithm is assumed to generate the candidate sets, *A* from the large itemsets. The count of *s(A)*, support of itemset *A* is obtained while scanning *T*.

- All single itemsets are candidates in the first pass. Any item with support value less than the specific minimum support is eliminated from the pool of candidate itemsets.
- The single itemsets are combined to form two members candidate itemsets. Support values of these candidates are then determined by scanning the database again. Same as before, candidates that has value less than minimum support will be eliminated in becoming frequent two itemsets.
- The next phase, candidates of three itemsets are created. This whole process stops only when all frequent itemsets are found and no further candidate itemset generation is possible.
- All these frequent itemsets are then used to generate association rules and only rules which satisfied the minimum confidence will be stored.

B. Defining Support Measure

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of *m* elements called items. A rule is defined as an implication of the form $X \rightarrow Y$, where $X, Y \subseteq I$ and $X \cap Y = \varnothing$.The left-hand side of the rule is named antecedent and the right-hand side is named consequent. Therefore, the association rules can be presented as below and the items are $\{i_2, i_4\}$:

$$i_2 \rightarrow i_4 \; [s = 40\%, conf = 60\%]$$

When a specific association satisfies the minimum support threshold, then *i* is identified as a frequent itemset.

Definition 1 (Support). Let $i \subseteq T$ be a set of items from database, *T*. The support of an itemset *i* in *T*, denoted by *s(i*), is the proportion of transactions that contain *i*, Eq. (1):

$$s(i) = \frac{\text{\# of transactions contains } i}{\text{\# of transactions}} \qquad (1)$$

If the support of an itemset *i* is more than minimum support, *minsup*, then *i* is a frequent itemset.

C. Apriori Algorithm Improved 1

Shirgaonkar *et al.* (2010) has implemented an application using the improved Apriori algorithm for book loan transactional database of a university library. This improvised version of Apriori is solely to increase the efficiency of Apriori in term of time taken for execution. Despite being a simple and easy algorithm, original Apriori algorithm suffers from vast and large generated number of candidates. This has led to highly cost of memory and time taken for each of the execution. The improved Apriori algorithm by Shirgaonkar *et al.* (2010) included a process to remove the transaction that do not have any frequent itemsets prior to mining process. This algorithm safely assumed the transaction/sample that does not have any frequent itemsets would not be considered as a frequent set. This follows the downward property in

Apriori algorithm that if the subset of the item is infrequent, so the superset cannot be frequent. The illustration of the improved algorithm is shown in Figure 2. The improved Apriori algorithm by Shirgaonkar *et al.* (2010) can be summarized as follows:

- All single itemsets are candidates in the first pass. Any item that has support values of less than specified minimum support is eliminated from the pool of candidate itemsets resulting in frequent one item set.
- For every row in transaction database where second item is 0 and first item is infrequent or second item is infrequent and third item is zero, delmark = 0 is marked. (Assume that the end of items in each row is marked with a zero).
- The single itemsets are combined to form two members candidate itemsets. Support values of these candidates are then determined by scanning the database again. Again, only the candidates above the pre-specified minimum support value are retained to get frequent two item set.
- For every row in transaction database where delmark = 1, scan it and determine if third item is 0 and first and second items are infrequent or fourth item is 0 and first and second items are infrequent or fourth item is 0 and second and third items are infrequent then mark delmark = 0.
- The next pass creates 3-member candidate itemsets and the process is repeated. This process stops only when all frequent itemsets are found and no further candidate itemset generation is possible.
- The frequent itemsets constitute the set of frequent items. These frequent item sets are then used to generate association rules which have confidence values greater than or equal to the specified minimum confidence values greater than or equal to the specified minimum confidence. Rules for frequent itemsets are then created.

Due to multiple scanning in the database, the input output equipment becomes heavy. The time taken is normally increasing the efficiency of the Apriori algorithm. However, when the algorithm removes the infrequent items from the original transactional database, the time taken for efficiency may be reduced.

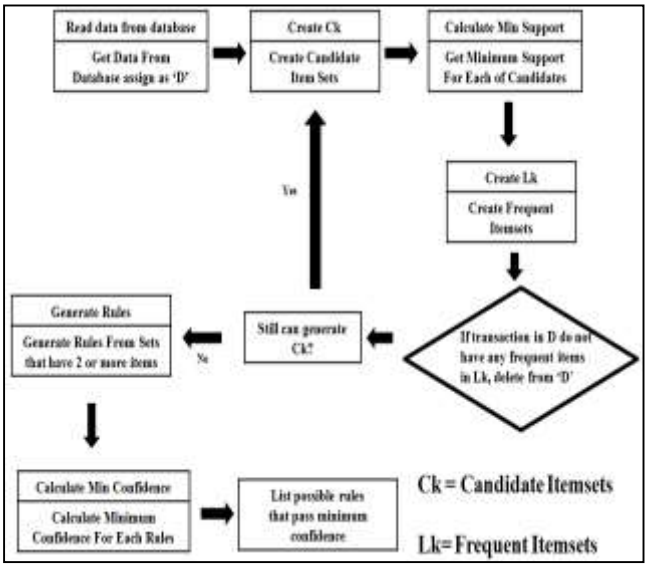

Figure 2. Flow of the Improved Algorithm 1.

D. Apriori Algorithm Improved 2

Another variant of Apriori was proposed by Kaur Gurneet (2014). The algorithm by Gurneet used a simple approach to decrease the time taken and memory used for the execution. This improvise version of Apriori used optimized method by reducing the size of database along with the number of itemsets generated, similarly to Shirgaonkar *et al.* (2010). Furthermore, this algorithm introduced a parameter, *SizeOfTransaction (SOT)*, that stored the number of items for each of the transaction. If the value of *SOT* matches the value of *k*, then the transaction will be deleted. Figure 3 shows the improved algorithm 2 and the pseudocode for this algorithm is shown as below (Gurneet, 2014).

- Firstly, *SOT* column is added to the database.
- In the first iteration, each item is a member of candidate 1-itemset, *C1*. The algorithm simply scans the database to count the occurrences of each item.
- The algorithm will then generate number of items in each transaction as a new parameter, namely; *SizeOfTransaction (SOT).*
- Depends on the minimum support, for example *min_supp* = 2, the set of frequent 1-itemset, *L1* can be determined.
- After *L1* were generated, the value of *k* becomes 2. Those records of transaction that have *SOT* = 1 in *T* were deleted. These records do not exist in any elements of *C2*.
- After that, the process is repeated until there is no candidate that can be generated.

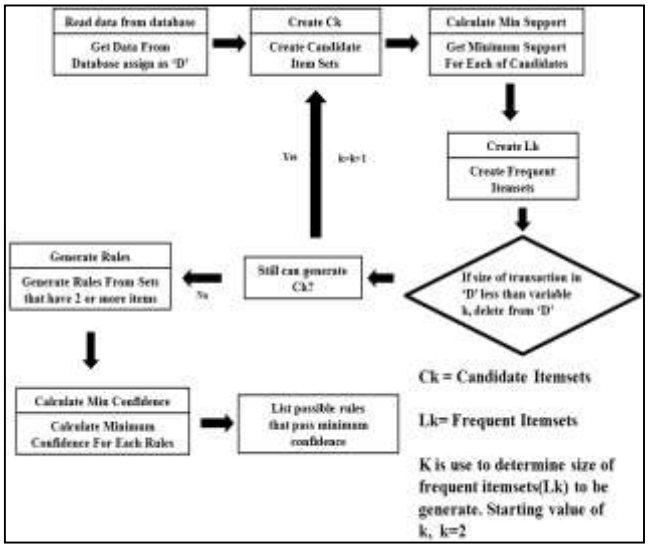

Figure 3. Flow of the Improved Algorithm 2

In order to find the efficient algorithm, we designed a comparative study of time execution in discovering all itemsets that are frequent. This particular method has been applied by various researches that interested in improving the frequent itemset mining algorithm (Bashir *et al*., 2006, Ahirwal *et al*., 2012, Yakop *et al*., 2015).

III EXPERIMENTAL SETUP

In this study, the comparative study was performed to investigate the original Apriori algorithms with two variants of Apriori algorithm, the improved Apriori algorithm by Shirgaonkar *et al.* (2010) as improved Apriori algorithm 1 and improved Apriori algorithm by Kaur Gurneet (2014) as improved Apriori algorithm 2. We were interested to know the outcome of these experiments, due to lack of result findings described in each paper. The experiment was done by varying minimum support value and number of transactions in dataset. The value of minimum support was adjusted and the number of scanning and time of execution were recorded. The numbers of transaction in database were set to 400, 600, 800 and 1000 respectively, and the minimum support are 0.1, 0.3, 0.5, 0.7 and 0.9. Table 2 shows the datasets used and Table 3 summarizes the testing on the algorithms. This experiment was conducted using Windows 8.1 64-bit operating system, Intel(R) Core i5 3.0Hz and 8.00GB RAM.

Table 2. Datasets Used in Experiments

Dataset	# of transactions	# of items
D1	400	1808
D2	600	2357
D3	800	2781
D4	1000	3182

Table 3. Testing of the Algorithms

Experiments	Parameter to be recorded
Different numbers of transaction	Time taken for each of the execution depends on scanning time.
Varying number of support value	Time taken for each of the execution depends on the minimum support use.

IV RESULTS

Figure 4-7 show the result of three Apriori based algorithms by using various support values between 0.1 and 0.9.

Figure 4. Time Taken for Different Support Value for D1.

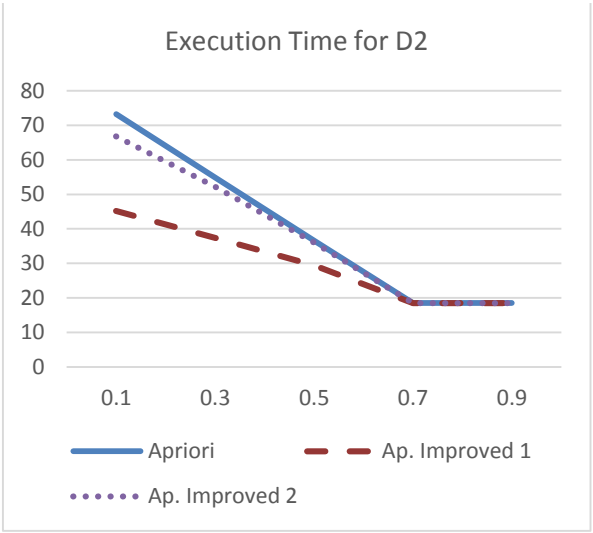

Figure 5. Time Taken for Different Support Value for D2

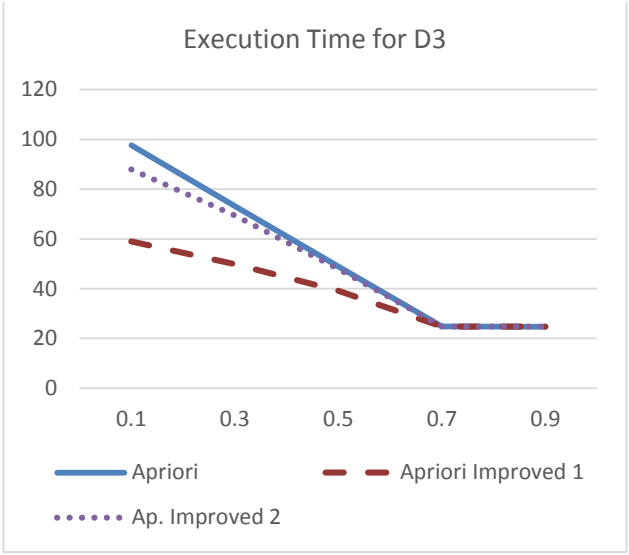

Figure 6. Time Taken Using Different Support Value for D3

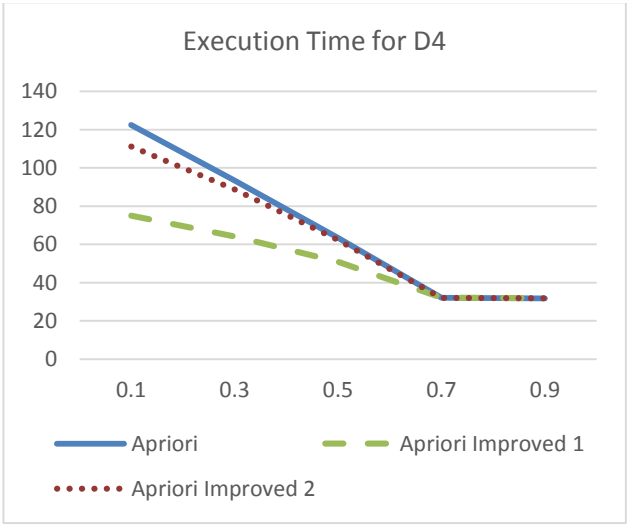

Figure 7. Time Taken Using Different Support Value for D4

As can be seen in all these four figures (Figure 4 until 7), there is an obvious difference in time taken for the execution of frequent itemset mining between the original Apriori and the improved Apriori algorithms. Original Apriori algorithm used more time for each of the execution compare to the improved Apriori algorithm. For example, when the experiment was carried out by using 400 transactions and 0.3 minimum support, original Apriori algorithm took 36.62 seconds, while improved Apriori algorithm 1 took 25.43 seconds and improved Apriori algorithm 2 took 35.07 seconds to be executed.

The comparison between Apriori Improved 1 and Apriori Improved 2 has shown that the latter algorithm seems need more time in finding the frequent itemsets. The Apriori Improved 1 algorithm shows a consistent behaviour with different set of transaction numbers. It shows that the strategy in deleting the item that less than the given support value from the transaction database could help the processing time. Scanning the new processed database without infrequent items does save a lot of time. Meanwhile, Apriori Improved 2 algorithm that uses a parameter that represent the number or items in each transaction, that is *SOT* does contributes the positive merit on processing time. This newly created parameter helps to find transaction that consist adequate item number.

V CONCLUSION

In this paper, the efficiency of the original Apriori algorithm and improved Apriori algorithms with various values of minimum support and number of transaction has been analysed. The results for each of the experiment have been recorded and comparisons have been made. Classical Apriori always need more time, as compared to the Improved Apriori algorithm. The improved algorithm 1 is outperfomed algorithm 2 in terms of less processing time and consistently giving good results. Further research using other improved Apriori algorithms on various parameters value can be done. There are several others improved Apriori algorithm (Mohammed & Bassam, 2014; Liao 2009) to be investigated, so that the good strategy can be always applied in proposing a new algorithm.

ACKNOWLEDGMENT

The authors would like to thank to the Ministry of Higher Education, Malaysia for the research grant FRGS 156/2013 and also to Research Management Centre, Universiti Teknologi MARA, Selangor, Malaysia for the support.

REFERENCES

Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. In J. B. Bocca, M. Jarke, & C. Zaniolo (Eds.), *Proc. 20th int. conf. very large databases, VLDB* (pp. 487–499). Morgan Kaufman. 12–15.

Aher, S. B., & Lobo, L. M. R. J. (2013). Combination of machine learning algorithms for recommendation of courses in E-Learning System based on historical data. *Knowledge-Based Systems*, 51, 1-14.

Ahirwal, R., Kori, N. K, and Jain, Y. K, (2012). Improved Data mining approach to find Frequent Itemset Using Support Count Table. *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)*, 1(2), 195-201.

Bashir, S., Shuaib, Sultan, M. Y, Baig, Rauf (2006). Improving Frequent Itemset Mining Algorithms. 2nd International Conference on Emerging Technologies. Pp 257-262.

Burke, R. (2002). Hybrid Recommender Systems: Survey and Experiments. User Modeling and User-Adapted Interaction, *Journal User Modeling and User-Adapted Interaction*, 12(4), pp. 331-370.

Borges, L. C., Marques, V. M., & Bernardino, J. (2013). Comparison of data mining techniques and tools for data classification. *Proceedings of the International C* Conference on Computer Science and Software Engineering*, 113-116. Doi:10.1145/2494444.2494451

Chen, Chia-Chen, & Chen, An-Pin. (2007). Using data mining technology to provide a recommendation service in the digital library. The Electronic Library, 25(6), 711-724. doi: doi:10.1108/02640470710837137

Chesnevar, C., Maguitman, A., & González, M. (2009). Empowering Recommendation Technologies Through Argumentation. In G. Simari & I. Rahwan (Eds.), Argumentation in Artificial Intelligence (pp. 403-422)

Chhavi Rana, S. K. J. (2012). Building a Book Recommender system using time based content filtering. *WSEAS Transactions On Computers*, Issue 2, Volume 11, February 2012, pp 27-33.

De Pessemier, T., Dooms, S., & Martens, L. (2013). Comparison of group recommendation algorithms. *Multimedia Tools and Applications*, 72(3), 2497-2541.

Kaur, G. (2014). Improving The Efficiency of Apriori Algorithm In Data Mining. *International Journal of Science, Engineering And Technology*, Volume 02 Issue 05 June- 2014, pp 315-326.

Han, J. and Kamber, M. (2006). *Data Mining: Concepts and Techniques*. 2nd Edition, Morgan Kaufmann Publishers.

LakshmiPriya, G., & Hariharan, Shanmugasundaram. (2012). An efficient approach for generating frequent patterns without candidate generation. Paper presented at the Proceedings of the International Conference on Advances in Computing, Communications and Informatics, Chennai, India.

Liao, B. (2009). An Improved Algorithm of Apriori. In Z. Cai, Z. Li, Z. Kang & Y. Liu (Eds.), *Computational Intelligence and Intelligent Systems* (Vol. 51, pp. 427-432)

Mohammed Al-Maolegi, B. A. (2014). An Improved Apriori Algorithm for Association Rules. Natural Language Computing. *International Journal on Natural Language Computing (IJNLC)* Vol. 3, No.1, February 2014, pp. 21-29

Mohammad-Arsyad, M.Y, Sofianita, M., Shuzlina Abdul-Rahman, Azlinah, M. (2015). Data Projection Effects in Frequent Itemsets Mining. Soft Computing in Data Science, Volume 545 of the series Communications in Computer and Information Science, 23-32.

Mettouris, C., & Papadopoulos, G. A. (2013). Ubiquitous recommender systems. *Computing*, 96(3), pp. 223-257.

Pazzani, M., & Billsus, D. (2007). *Content-Based Recommendation Systems*. In P. Brusilovsky, A. Kobsa & W. Nejdl (Eds.)

Rajola, F. (2013). Data Mining Techniques Customer Relationship Management in the Financial Industry (pp. 109-125)

Resnick, P. and Varian, H.R. Recommender Systems. *Communications of the ACM*, Vol. 40, No. 3, 1997, pp. 56–58

Shirgaonkar, S., Rajkumar, T., & Singh, V. (2010). Application of improved Apriori in University Library. *Proceedings of the International Conference and Workshop on Emerging Trends* in Technology, Mumbai, Maharashtra, India. Pp 535-540, do:>10.1145/1741906.1742027

Xujing, B., & Weixiang, X. (2013). The Research of Improved Apriori Algorithm. In Z. Zhang, R. Zhang & J. Zhang (Eds.), LISS 2012 (pp. 1007-1012): Springer Berlin Heidelberg.