Determining the maximum number of transaction records that the Apriori algorithm can scan in 90 seconds

CHRISTIAN DALE P. CELESTIAL, BASSY LEIAH V. IBARRETA, RUTH SP G. TIRON, MARIA MILAGROSA A. NULLA, and ZENNIFER L. OBERIO

Philippine Science High School - Western Visayas Campus, Brgy. Bito-on, Jaro, Iloilo City 5000, Department of Science and Technology, Philippines

Abstract

The Apriori algorithm is a data mining algorithm used for frequent itemsets. It is easy and simple to use, but its main disadvantage is its inefficiency in scanning large databases. Studies about the algorithm focus on improving its efficiency in large databases, but there is no definite value yet as to the maximum number of transactions that the Apriori algorithm can process in 90 seconds, the tolerable offline waiting time for the human attention span. The methods of this study consist of the hardware and database acquisition, program implementation, data collection, and data analysis. Five hundred transactions were first scanned using the algorithm. It was determined that the classic Apriori algorithm can process 1,310 transaction records in 90 seconds, with a percentage prediction error of 0%. The percentage prediction error was computed using the actual and outputted frequencies.

Keywords: *Apriori algorithm, data mining, frequent itemsets, percentage prediction error, accuracy*

Introduction. Data mining is a process that analyses large databases in order to discover meaningful patterns [1]. The Apriori algorithm is a classic data mining algorithm for frequent itemset mining. It is under the association rule technique of data mining which was initially introduced by Rakesh Agrawal [2]. It enumerates all of the frequent itemsets in a database [3] and is considered best to be used for closed itemsets [2].

Although the algorithm is easy and simple to use, its main disadvantage is that it is not suitable for large databases because its performance declines as the number of transaction records increase. When the database contains a large number of transaction records, scanning the database for frequent itemsets becomes time-consuming [4]. Previous studies pointed out that the algorithm needs to scan the database several times [5], that it is limited to only a small database [6], and that the time that it takes for the algorithm to scan the database increases as its size increases [7]. The performance of the algorithm in dense data is also shown to decline due to the large number of long patterns [8]. Aggarwal and Sindhu [9] discovered that the Apriori algorithm works inefficiently in terms of memory requirement when large numbers of transaction records are considered.

According to the study by Al-Maolegi and Arkok [10], the Apriori algorithm has two parameters to consider, namely the minimum support and confidence level, which are both set by the user. The scanning time of the algorithm is affected by the minimum support because it indicates the number of itemsets to be scanned by the algorithm. The minimum support is used to exclude itemsets in the results which have a support less than the set minimum support. The support of an itemset is the number of transactions that contain all the items of that itemset. A small minimum support would mean that a large number of itemsets will be considered in the scanning process whereas, a large minimum support would mean that the algorithm would be considering only few itemsets.

Studies about improving the algorithm's performance focus on improving its scanning time and accuracy. Different solutions and improvements such as the Bit Array Matrix by Vijayalakshmi and Pethalakshmi [7] improved the algorithm's scanning time and accuracy in large databases. Other solutions are those by Kaur [8] which based the improvement of the algorithm on the accuracy alone, and the studies by Rehab *et al.* [4], Singh *et al.* [11], and Najadat *et al.* [12] which improved the algorithm's scanning time only.

Although some studies have already described the number of transaction records scanned by the algorithm and the algorithm's scanning time [6,12], there is no definite maximum value yet as to the number of transaction records that can be scanned in 90 seconds, the tolerable offline waiting time for the human attention span. [13].

This study aimed to determine the size of the database which Apriori can process accurately at a tolerable waiting time, which is 90 seconds, given that the complexity of the database and the hardware used are constant. Specifically, it aimed to:

(i) determine the algorithm's scanning time and accuracy in a database with an increasing number of transaction records; and

(ii) determine the maximum number of transaction records that the algorithm can process within the tolerable waiting time of 90 seconds, with a percent error of 0%.

How to cite this article:
CSE: Celestial CDP, Ibarreta BLV, Tiron RSG, Nulla MMA, Oberio ZL. 2020. Determining the maximum number of transaction records that the Apriori algorithm can scan in 90 seconds. Publiscience. 3(1):128-131.
APA: Celestial, C.D.P., Ibarreta, B.L.V., Tiron, R.S.P., Nulla, M.M.A., & Oberio, Z.L. (2020). Determining the maximum number of transaction records that the Apriori algorithm can scan in 90 seconds. *Publiscience, 3*(1):128-131.

The results of this study will benefit researchers who aim to further enhance the performance of the Apriori algorithm. The number of transactions scanned by the enhanced algorithm can be compared with the quantified results from this study.

Methods. This study aimed to determine the size of the database which Apriori can process accurately in 90 seconds, given that the complexity of the database and the hardware used are constant. Specifically, it aims to determine the algorithm's scanning time and accuracy in a database with an increasing number of transaction records. It also aims to determine the maximum number of transaction records that the algorithm can process within 90 seconds [13], with a percent error of 0%.

Hardware and Database Procurement. A laptop with an Intel Core i5 3.4GHz Processor with 4GB RAM was used in the testing process. A grocery database having 9,835 transaction records in total was acquired from the website of Salem Marafi [14]. It contained a collection of receipts with each line representing one (1) receipt and the items purchased. The database was converted from a comma-separated values (.csv) file to a text (.txt) file by replacing the commas with spaces, since the source code used required plain text for its input.

Program Implementation. The Java source code of Apriori was acquired from Github [15] and was modified to output the execution time of the algorithm and the frequency of the frequent itemsets. The modified source code was then checked by two consultants who had a background on data mining.

Data Collection. The testing process had three trials for each set of transactions, with the execution time and frequency tabulated after each set of trials. Five hundred transaction records were initially used during the testing process, based on the study of Najadat *et al.* [12], with an increment of 200 transaction records added after each set of trials, following the methods of Sahu *et al.* [6]. The process was repeated until the average scanning time of the algorithm exceeded 90 seconds [13], after which the transaction records added were reduced by half until the scanning time reached 90 seconds.

A minimum support of five was used throughout the whole testing process.

It must be noted that the Java IDE used in the study was closed after each trial to ensure that the program's memory consumption did not interfere with the algorithm's scanning time.

Data Analysis. The average scanning time of the algorithm was first graphed using a scatter chart in Microsoft Excel. A trendline was then added.

The percentage prediction error (PPE) was analyzed using the outputted frequency (OF) and actual frequency (AF). It was computed using the equation below [16],

$$PPE = \frac{(OF - AF)}{OF} \; x \; 100$$

Using the frequency displayed by the program, which served as the outputted frequency, and the actual frequency of the frequent itemsets, the percentage prediction error was determined.

In order to determine the actual frequency of each itemset, the transaction records from the database were transferred to a word document. Next, the itemsets that were displayed by the program were used as a guide for the manual searching of each item contained in the frequent itemset. The items contained in the itemset were searched individually to remove the transaction records which did not contain the items. This was repeated until all the items in the frequent itemset were located in the database. The remaining transaction records contained all the items in the frequent itemset, and these transaction records were counted manually. The same process was repeated for all the frequent items displayed by the program. The number obtained served as the actual frequency, which was used in the calculation for the percentage prediction error of the algorithm.

Safety Procedure. Ensuring the privacy of users' data and the integrity of the data was a key ethical issue which the group took into account. The code used and modified for the purpose of this study was cited, acknowledging the creator of the said code. Proper credit was also given to the source of the database used in the study.

Results and Discussion. Table 1 shows the average scanning time for each set of transaction records. It can be observed that the scanning time of the algorithm increased between 1,100 and 1,300 transaction records, where the scanning time rose from 15.25 seconds to 77.121 seconds. The maximum number of transaction records that the algorithm processed under 90 seconds can also be seen in Table 1 – in this case, the algorithm was able to process 1,310 transaction records. The average scanning time of the algorithm for 1,310 transaction records was 82.53 seconds.

Table 1. The average scanning time of each set of transaction records.

Number of Transaction Records	Average Scanning Time (s)
500	0.536
700	1.756
900	4.953
1100	15.25
1300	77.121
1310	82.53
1311	91.57
1350	103.238
1400	117.551
1500	148.69

This increase in the scanning time of the algorithm can also be seen in the graph in Figure 1, where it can be observed the increase was between 1,100 to 1,300 transaction records. This is also seen in the trendline of the graph in Figure 1, as the average scanning time is seen to rise exponentially as the number of transaction records increase.

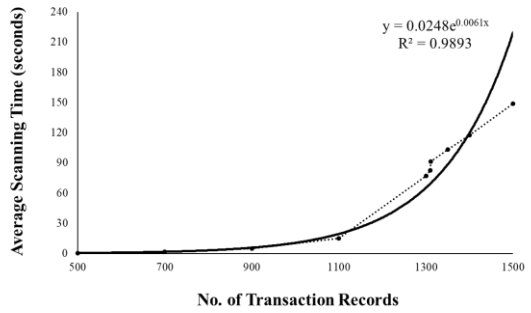

Figure 1. The average scanning time of the algorithm and the corresponding number of transaction records.

The accuracy of the algorithm was computed using the equation for percentage prediction error. The percentage prediction error of the first itemsets in every set of transaction records was 0% as shown in Table 2.

Table 2. Percent error of the algorithm computed using the actual and outputted frequency.

Number of Transaction Records	Actual Frequency	Outputted Frequency	Percent Error (%)
500	5	5	0
700	5	5	0
900	5	5	0
1100	6	6	0
1300	5	5	0
1310	5	5	0
1311	5	5	0

The results show that the highest number of transaction records that the classic Apriori algorithm can process under 90 seconds is 1,310 transaction records. The average scanning time of the algorithm in scanning 1,310 transaction records was 82.53 seconds.

The time complexity of the classic Apriori algorithm is $O(2^n)$ [17]. Exponential (base 2) running time means that the calculations performed by an algorithm double every time as the input grows. This can be supported by Figure 1 due to the exponential trendline having an $R^2 = 0.9893$ as compared to other types of trendlines which have a lesser R^2 value.

In the study by Vijayalakshmi and Pethalakshmi [18], the classic Apriori algorithm was able to scan about 1,750 transactions in under 0.12 seconds. In the current study, it can be observed that the algorithm scanned 1,100 transaction records in 15.25 seconds, which shows that the algorithm was able to scan less transaction records in a greater amount of time. The difference in the number of transaction records scanned by the algorithm may have been due to the difference of the databases used by the researchers. A numerical database with 1's and 0's was used in the study by Vijayalakshmi and Pethalakshmi [18], while a grocery database, composed of strings, was used in this study.

Another variable that may have caused the differences between the studies would be the minimum support. The performance of the algorithm is strongly dependent on its minimum support [19], since having a lesser minimum support would make the algorithm more flexible in accepting associations, requiring more time for it to process.

In relation to this, the database had varying number of items in the itemsets, which could have affected the scanning process of the algorithm. This is because having lesser number of items in an itemset would reduce the number of transaction records for an association. Thus, fewer associations will be considered, due to having a support less than the minimum support [8].

The results of this study could be used as a basis for researchers who aim to study the same algorithm in the future. Since the classic Apriori algorithm is still being improved [7], researchers may use the results of this study to compare the scanning time of the classic Apriori algorithm with their improved version of the algorithm. For example, if the number of transaction records scanned by their improved algorithm surpasses 1,310 records, then it can be determined that the improved Apriori algorithm is effective. The trendline acquired in the study could be used to predict the increase of the scanning time of the algorithm during the testing process, given the same parameters.

The methods of this study could be adopted by researchers and compare their results with this study to aid in the improvement of the classic Apriori algorithm, provided that the database, minimum support, and processor used will be similar to those used in this study.

Limitations. The limitations of this study include its applicability to the Java source code of the classic Apriori algorithm. the varying number of items in each itemset of the database and the inefficiency of the testing process.

If the number of items in the itemset is lesser, the number of transaction records for an association would be lesser as well. This leads to fewer associations being considered, due to having support less than the minimum support [8].

The hardware used in the testing process could have also limited the number of transaction records scanned because better hardware would be able to scan more transaction records in a short amount of time according to Aho et al. [20].

The inefficiency of the testing process is a also a limitation since the source code used could not read text files automatically. The contents of the database had to be copied and pasted to the source code, which consumed an additional two minutes for each trial.

Conclusion. Although the Apriori algorithm is easy and simple to use, its main disadvantage is its inefficiency in scanning large databases. Studies performed on the algorithm have focused on improving it, and some studies have already described the number of transaction records scanned by the algorithm and the algorithm's scanning time. There is, however, no definite maximum value yet as to the number of transaction records that can be

scanned in a tolerable amount of time, which is 90 seconds. The results of the study showed that this value is 1,310 transaction records with a percentage prediction error of 0% throughout the whole process. The factors that affect the scanning time of the classic Apriori algorithm are the processor of the hardware and the minimum support. These variables were kept constant throughout the whole study. This study can aid the improvement of the algorithm by using the results as a basis for future studies.

Recommendations. In order to improve this study, it can be performed using other types of Apriori algorithms. It can also be tested using the source code of a different programming language such as C++ and Python. Other types of hardware can be used in the testing process to determine the hardware's effect on the scanning time of the algorithm. It is also recommended to redesign the used source code to be able to process a database in a text file rather than manually copying and pasting items from the database to the source code. For future studies, the columns of the database must be filled out when testing the algorithm. There must be no null value in any column of the database so that the number of items in each transaction record will be consistent. Through this, the minimum support will be applicable to all the transaction records.

Acknowledgement. The group would like to extend their gratitude to Mr. Christian Chiu and Mr. Marc San Pedro for taking the time to evaluate the group's modified source code.

References

[1] Neha D and Vidyavathi BM. 2015. A survey on applications of data mining using clustering techniques. Int J Comput Appl. 126(2): 7-12.

[2] Prithiviraj P and Porkodi R. 2015. A comparative analysis of association rule mining algorithms in data mining: A study. Am J Comput Sci Eng Surv. 3(1): 98-119.

[3] Yabing J. 2013. Research of an improved Apriori algorithm in data mining association rules. Int J Comput Comm Eng. 2(1): 25-27.

[4] Rehab A, Alwa H, Patil AV. 2013. New matrix approach to improve Apriori Algorithm. Int J Comput Sci Netw Soln. 1(4): 102 -109.

[5] Wei Y, Yang R, Liu P. 2009. An improved Apriori algorithm for association rules mining. Proceedings of a symposium held at the 2009 Institute of Electrical and Electronics Engineers International Symposium on IT in Medicine an Education; Jinan, China.

[6] Sahu A, Dhakar M, Rani P. 2015. Comparative analysis of Apriori algorithm based on association rule. Int J Comp Sci Comm. 6(2):18-21.

[7] Vijayalakshmi V and Pethalakshmi A. 2013. Mining of frequent itemsets with an enhanced Apriori algorithm. Int J Comput Appl. 81(4): 1-4.

[8] Kaur G. 2014. Improving the efficiency of Apriori algorithm in data mining. Int J Sci Eng Tech. 2(5): 315-326.

[9] Aggarwal S and Sindhu R. 2015. An approach to improve the efficiency of the Apriori algorithm. PeerJ PrePrints [Internet]. [cited 2018 Nov 21]; 1-13. Available from: https://doi.org/10.7287/peerj.preprints.1159v

[10] Al-Maolegi M and Arkok B. 2014. An improved Apriori algorithm for association rules. Int J Nat Lang Comp. 3(1): 21-29.

[11] Singh J, Ram H, Sodhi JS. 2013. Improving efficiency of Apriori algorithm using transaction reduction. Int J Sci Res Pub. 3(1): 1-4.

[12] Najadat HM, Al-Maolegi M, Arkok B. 2013. An improved Apriori algorithm for association rules. Int Res J Comput Sci Appl. 1(1): 1-8.

[13] Antonides G, Verhoef PC, van Aalst M. 2002. Consumer perception and evaluation of waiting time: A field experiment. J Consum Psychol. 12(3): 193–202.

[14] Salem. 2014. Market basket analysis with R [Internet]. Salem Marafi [cited 2019 Sept 13]. Available from: http://www.salemmarafi.com/code/market-basket-analysis-with-r/

[15] Umanghome. 2016. Apriori implementation in Java [Internet]. Github Gist [cited 2019 Sept 9]. Available from: https://gist.github.com/umanghome/2c1ee7f03eb99dc9fde2512be3fd36fd

[16] Wu G, Baraldo M, Furlanut M. 1995. Calculating percentage prediction error: A user's note. Pharmacological Res. 32(4): 241-248.

[17] Pai GAV. 2008. Data structures and algorithms: Concepts, techniques, and applications. India: McGraw Hill Education, 12-19 p.

[18] Vijayalakshmi V and Pethalakshmi A. 2015. An efficient count based transaction reduction approach for mining frequent patterns. 47(2015):52-61.

[19] Mehta D and Samvastar M. 2017. Implementation of improved Apriori algorithm on large dataset using Hadoop. Asian J Comput Sci Eng. 2(6): 10-13

[20] Aho AV, Hopcroft JE, Ullman JD. 1974. The design and analysis of computer algorithms. Reading (MA): Addison-Wesley. 2