Communications in Mathematics and Applications
Vol. 14, No. 3, pp. 1143–1152, 2023
ISSN 0975-8607 (online); 0976-5905 (print)
Published by RGN Publications
DOI: 10.26713/cma.v14i3.2375

http://www.rgnpublications.com

Special Issue
Recent Trends in Mathematics and Applications
Proceedings of the International Conference of
Gwalior Academy of Mathematical Sciences 2022
***Editors*: Vinod P. Saxena and Leena Sharma**

Research Article

Analyze the Novel Apriori Algorithm for Frequent Pattern Generation and Improving the Performance by Generating Powersets for Unique Records

Sagar Lahade[1], Afrin Ali[1], Rushikesh More[1], Sanjay Mahajan*[1], Kannan Rajeswari[1], Sushma Vispute[1], Reena Kharat[1] and N. Vivekandandan[2]

[1] *Department of Computer Engineering, Pimpri Chinchwad College of Engineering, Pune, Maharastra, India*
[2] *Department of Mechanical Engineering, Pimpri Chinchwad College of Engineering, Pune, Maharastra, India*
***Corresponding author:** sanjaymmahajan@gmail.com

Received: February 8, 2023 **Accepted:** June 20, 2023

Abstract. Knowledge discovery is the course of absorbing facts and relations from immense amounts of data. Algorithms like Apriori let an experimenter locate the hidden pattern within a dataset. However, numerous applications do not utilize the Apriori technique because it takes a prolonged time to discover the frequent itemset. If the largest frequent itemset with length k exists, the algorithm accomplishes a k scan to deal with the time-consuming difficulty caused by the k number of scans. Rajeswari and Vaithiyanathan (A novel method for frequent pattern mining, *International Journal of Engineering and Technology* **5**(3) (2013), 2150 – 2154) devised a Novel algorithm for frequent pattern mining that uses a single scan to discover a frequent itemset of length k by constructing a subset of transactions. In this work, the Novel algorithm's performance is embellished by generating powersets for unique records. Powerset generation is a costly operation that takes exponential time to compute. So, avoidance of unnecessary computation results in performance enhancement. To accomplish used two mechanisms to improve the performance of the Novel method viz *Transaction Subset Plus Cache* (TSPC) and *Transaction Subset Without Duplicate Record* (TSWDR). Finally, performance analysis is done between four algorithms Apriori for frequent pattern generation, Novel method for frequent pattern generation, TSPC and TSWDR.

Keywords. Apriori, Frequent pattern

Mathematics Subject Classification (2020). 05A15

1. Introduction

In a transactional database, the challenge of creating large frequent itemset for the development of association rules is discussed. Numerous algorithms such as algorithms, such as Apriori, FP growth, and their variants, have already been proposed in this subject. To find association rules between items, classic Apriori algorithms involve two major phases: (a) frequent pattern generation, (b) rules creation.

Several applications, however, are unable to employ the Apriori approach because of the lengthy time it takes to discover the common itemset and build rules. Most of the time is spent in Apriori looking for common patterns. By establishing a power set of items present in transactions, in [11] Rajeswari and Vaithiyanathan suggested a novel way for computing frequent itemset with a single database scan. With a single database scan, their method can yield all levels of the candidate set as well as the support count. In the case of the Apriori approach, we need to search the database k times if there exists a frequent itemset of length k in the given dataset for the given lowest support. As the number of scans grows, so does the computation time. The time necessary to calculate the frequent pattern grows as the number of scans increases, i.e., the time required by the conventional Apriori technique for discovering a frequent itemset is equal to the time required for a single scan multiplied by k.

The following paragraph confers some of the elements that influence the Apriori algorithm's complexity. Because the number of database checks essential to identify frequent item sets grows exponentially as the number of attributes rises, the time required to construct the nth frequent itemset rises exponentially. The average transaction width in a dense relational database can be quite significant. Because more computation time is needed to handle the dense transactions, this influences the Apriori algorithm's complexity.

Itemset will be developed during the candidate generation process where n signifies the number of different items in the dataset and k the length of each itemset.

2. Equations

2.1 Existing Method

Rajeswari and Vaithiyanathan [11] devised a method for locating frequent itemset of any length that does not generate candidate sets. They generated frequent itemset with a single database scan by using a power set of items present in a particular transaction of transactional database D. Suppose database D contains n transactions $\{p_1, p_2, p_3, \ldots, p_n\}$, each transaction p_i is a subset of $i = \{i_1, i_2, i_3, \ldots, i_n\}$ i.e., i is a universal set of items in the dataset. To find frequent itemset of length k example and algorithm of novel frequent pattern mining using powerset [11] given below:

Step 1: Examine the database D.

Step 2: For per transaction, p_i contains a width 'wth' (number of items) where $p_i \in D$ perform below step:

- Record all the 2^{w-1} combinations i.e., (powerset of items present in transactions except empty set) in Table 1 and increase the occurrence count for each element of power set.

- If the element e is already available in the list, increment its occurrence counts.
- Otherwise, create a new record in Table 1 and set its occurrence count to 1.
- Sort data in descending order based on the length of itemset and finally select itemset, which qualifies minimum support criteria.

Table 1. Example given dataset

Transaction ID	Itemset
1	A, B, C
2	B, D
3	A, B, C

Table 2. Proposed algorithms

Transaction 1: A, B, C		
Itemset	Occurrence Count	Length of Itemset
A	1	1
B	1	1
C	1	1
A, B	1	2
A, C	1	2
B, C	1	2
A, B, C	1	3

Table 3. Second transaction table

Transaction 2: B, D		
Itemset	Occurrence Count	Length of Itemset
A	1	1
B	$1+1$	1
C	1	1
A, B	1	2
A, C	1	2
B, C	1	2
A, B, C	1	3
D	1	1
B, D	1	1

Table 4. Third transaction table

Transaction 3: A, B, C		
Itemset	Occurrence Count	Length of Itemset
A, B, C	2	3
D	1	1
B, D	1	1

Table 5. Sorting transaction table

Sort according to itemset length		
Itemset	Occurrence Count	Length of Itemset
A, B, C	2	3
A, B	2	2
A, C	2	2
B, C	2	2
A	2	1
B	3	1
C	2	1
D	1	1
B, D	1	1

Table 6. Transaction with min_supp= 2

Minimum Support = 2		
Itemset	Occurrence Count	Length of Itemset
A, B, C	2	3
A, B	2	2
A, C	2	2
B, C	2	2
A	2	1
B	3	1
C	2	1
D	1	1
B, D	1	1

Analysis of the Existing Method. This method determines the transaction's power set. Calculating a powerset is a time-consuming process. Calculating the powerset of a set containing n items requires $O(2^n)$ time. As a result, the time it takes to calculate powerset for 'm' transactions is $(m * 2^{n/2})$. It requires less time than performing k scans to locate a frequent itemset in Apriori, which takes $O(2^d)$, where d is the number of unique items in the database.

2.2 Proposed Method

To give rise to frequent item sets, a novel method for frequent pattern generation takes $O(m * 2^{n/2})$. The entire number of transactions in the database is represented by 'm'. Calculating the power set for unique transactions only reduces time complexity. The same result was produced by using the value of several repetitions(cache), as well as by eliminating duplicate rows from the dataset and creating a single column that reflects the number of times the transaction occurred. By multiplying the number of repetitions by the occurrence count of each element of a transaction's powerset.

As a result, the temporal complexity of the new method was decreased to $(u * 2^{n/2})$ where 'u' denotes the number of distinct transactions.

Table 7. Existing method algorithms

Apriori Algorithm
1. Assume that k is equal to one
2. Identify the most common L_1-length itemset
3. Continue until no new frequent itemset are discovered
4. From the length k-frequent itemset, create a length $k+1$ candidate itemset
5. Prune infrequent subgroups of length k from the candidate itemset
6. Scan the database production to determine each candidate's support
7. Eliminate infrequent candidates and maintain only those who are regular
A Two-Step Process
1. *Create itemset on a regular basis* – Create all itemset whose support is less than minimum support
2. *Formulation of rules* – Create high-confidence rules for each often-occurring item collection

Novel Method
1. Make an FP map to keep track of objects and their frequency of recurrence
2. Continue with the remaining rows (a) Take a look at a database record (b) Make a 'S' set for items that appear in a row (c) Make a 'P' power set for 'S' (d) Verify whether each set of 'P' is already present in FP or not If yes, increase the number of times it occurs. If no, enter it into the FP with an occurrence count of one
3. Select elements from FP that have an occurrence count more than or equal to the lowest support
4. From an individually common item set, create high confidence rules

As a result, the Novel technique dramatically lowered the time complexity of generating frequent patterns. The authors, however, have discovered that the Novel technique generates a power set for each row. This can be avoided by eliminating all duplicate entries from a dataset and simply adding a 'repeated count' column. It takes more time to produce a powerset. Various permutations and combinations of items must be performed to produce a power set algorithm. The number of elements/attributes present in the transaction is exactly commensurate to the number of elements/attributes present in the transaction. For power set creation, it is obvious that the time complexity is $O(2^n)$ and the space complexity is $O(2^n)$.

3. Algorithm

So instead of calculating the power set for all rows authors used the technique so that this calculation is done for unique rows/transactions only. To accomplish this, they used two methods.

(1) Removing duplicate rows from the dataset and adding one more column for the repeat count in such a way that the result will not get affected (TSWDR). The time complexity of grouping rows is $O(n)$ if data is sorted then it gets reduced to n $(\log n)$.

(2) By creating a cache-like system (novel method with additional storage) in this method (TSWDR) algorithm calculates the power set for a row and stores it into the cache. So, whenever the same row came for computation of the power set. It first checks whether a precalculated powerset is present in the cache for this row. If yes, then return the powerset from the cache. If not, calculate the power set and store it into the cache for future use.

Table 8. Proposed algorithm

Novel Method With Cache (TSPC)
1. Create map FP to store items with their occurrence count and map 'PS' to store powerset
2. Repeat for all rows
(a) Read database row
(b) Create a set 'S' for items, which are present in row
(c) Check if the power set of 'S' is available in 'PS' or not if yes then get the powerset from PS. If not, then compute the power set and store it in PS
(d) Get every set of the power set and increment its occurrence count of that set in FP
3. Select only, sets that have occurrence count ≥ min support

Novel Method After Removing Duplicate Rows (TSWDR)
1. Create map FP to store items with their occurrence count
2. Remove all duplicate rows using the panda's library and add an additional column that will hold a repeated count of row
3. Repeat for all rows
(a) Read database row
(b) Create a set 'S' for items, which are present in row
(c) Generate power set 'P' of 'S'
(d) Get every set of the power set 'P' and update its occurrence count with previous occurrence Count repeated row count
4. Select only, sets that have occurrence count ≥ min support

The authors then compared the performance of these four algorithms in various scenarios such as unique rows, duplicate rows with only a few frequent items among all attributes, and so on. Following that, we reach the following conclusions.

(1) When the dataset contains more unique rows then the Novel Apriori algorithm is the best choice.

(2) When repeated rows are present in large quantities, Novel Apriori with removing duplicate rows (TSWDR) or by storing an already calculated power set into cache like system (TSPC), we can improve the algorithm.

(3) If a single element meets the minimum support criteria in the candidate set 1, normal Apriori will provide the quickest solution among algorithms, but Novel Apriori will remove duplicate rows. Because the database does not need to be rescanned in the Novel Apriori /Novel Apriori cache, we must generate a powerset, which takes longer.

Results

Table 9. Comparison table

ANALYSIS						
Repeated count	Rows	Attributes	Algorithm	Min support	Total time required	Remark
0	30	5	Novel Apriori	0.2	32.6	
0	30	5	Novel Apriori with additional storage (Cache to get precalculated powerset) to avoid repeated work (TSPC)	0.2	36.6	
0	30	5	Novel Apriori after removing duplicate rows using panda (TSWDR)	0.2	35.8	
0	30	5	Apriori	0.2	78.4	
0	7	3	Novel Apriori	0.5	18	
0	7	3	Novel Apriori with additional storage (Cache to get precalculated powerset) to avoid repeated work (TSPC)	0.5	18	
0	7	3	Novel Apriori after removing duplicate rows using panda (TSWDR)	0.5	18.1	
0	7	3	Apriori	0.5	25.4	
0	99	8	Novel Apriori	0.1	124	
0	99	8	Novel Apriori with additional storage (Cache to get precalculated powerset) to avoid repeated work (TSPC)	0.1	143	
0	99	8	Novel Apriori after removing duplicate rows using panda (TSWDR)	0.1	161	
0	99	8	Apriori	0.1	468	
125	129	3	Novel Apriori	0.7	170	
125	129	3	Novel Apriori with additional storage (Cache to get precalculated powerset) to avoid repeated work (TSPC)	0.7	133	

Table *Contd.*

ANALYSIS						
Repeated count	Rows	Attributes	Algorithm	Min support	Total time required	Remark
125	129	3	Novel Apriori after removing duplicate rows using panda (TSWDR)	0.7	33.6	
125	129	3	Apriori	0.7	233	
868	875	11	Novel Apriori	0.7	19400	
868	875	11	Novel Apriori with additional storage (Cache to get precalculated powerset) to avoid repeated work (TSPC)	0.7	11600	
868	875	11	Novel Apriori after removing duplicate rows using panda(TSWDR)	0.7	254	
868	875	11	Apriori	0.7	1030	Only one candidate set of size 1 qualify min support
868	875	11	Novel Apriori	0.1	12400	
68	75	1	Novel Apriori with additional storage (Cache to get precalculated powerset) to avoid repeated work (TSPC)	0.1	11100	Because no of elements present in each row is less than 6, so time required to generate powerset and storing it in dictionary (cache) is same
68	75	1	Novel Apriori after removing duplicate rows using panda (TSWDR)	0.1	354	
68	75	1	Apriori	0.1	0000	

Table 10. Parameter comparison

Serial Number	Parameter	Apriori Algorithm	Novel Apriori Algorithm	TSPC and TSWDR
1	Number of combination generated	t*2**n	t*2**(n/2)	u*2**(n/2)
2	Number of comparisons	t*2**n*k	t*k	u*k
3	Number of passes	2**n	1	1

Notes: t = number of transactions,
n = number of items,
k = number of frequent itemset or stages,
u = number of unique records/rows

4. Conclusions

The authors discovered that the powerset generation step in the novel method for frequent pattern generation consumes the most time because power set generation has space and time complexity $O.(2n)$. Based on that observation, the authors created an algorithm that prevents the generation of power sets from being repeated for identical/duplicate rows. They then compared performance and discovered that when a dataset contains many duplicate rows, the performance of the novel method for frequent pattern generation is significantly improved after generating a powerset for unique transactions only, which is normal in the case of an Apriori algorithm.

Competing Interests

The authors declare that they have no competing interests.

Authors' Contributions

All the authors contributed significantly in writing this article. The authors read and approved the final manuscript.

References

[1] R. Agrawal and R. Srikant, Fast algorithms for mining association rules, in: *Proceedings of the 20th International Conference on Very Large Data Bases* (VLDB'94), Vol. 1215, 1994, pp. 487 – 499, URL: https://www.vldb.org/conf/1994/P487.PDF.

[2] S. Chai, J. Yang and Y. Cheng, The research of improved Apriori algorithm for mining association rules, in: *Proceedings of 2007 International Conference on Service Systems and Service Management*, Chengdu, China, 2007, pp. 1 – 4, DOI: 10.1109/ICSSSM.2007.4280173.

[3] Y. Cong, Research on data association rules mining method based on improved Apriori algorithm, in: *Proceedings of 2020 International Conference on Big Data & Artificial Intelligence & Software Engineering* (ICBASE), Bangkok, Thailand, 2020, pp. 373 – 376, DOI: 10.1109/ICBASE51474.2020.00085.

[4] M. Dehghani, A. Kamandi, M. Shabankhah and A. Moeini, toward a distinguishing approach for improving the Apriori algorithm, in: *Proceedings of 2019 9th International Conference on Computer and Knowledge Engineering* (ICCKE), Mashhad, Iran, 2019, pp. 309 – 314, DOI: 10.1109/ICCKE48569.2019.8965206.

[5] J. Du, X. Zhang, H. Zhang and L. Chen, Research and improvement of Apriori algorithm, in: *Proceedings of 2016 Sixth International Conference on Information Science and Technology* (ICIST), Dalian, China, 2016, pp. 117 – 121, DOI: 10.1109/ICIST.2016.7483396.

[6] Y. Jia, G. Xia, H. Fan, Q. Zhang and X. Li, An improved Apriori algorithm based on association analysis, in: *Proceedings of 2012 Third International Conference on Networking and Distributed Computing*, Hangzhou, China, 2012, pp. 208 – 211, DOI: 10.1109/ICNDC.2012.56.

[7] N. Karimtabar and M. J. S. Fard, An extension of the apriori algorithm for finding frequent items, in: *Proceedings of 2020 6th International Conference on Web Research* (ICWR), Tehran, Iran, 2020, pp. 330 – 334, DOI: 10.1109/ICWR49608.2020.9122282.

[8] J. Liu and M. Cong, The optimization of apriori algorithm based on array and its application in the analysis of insurance clients, in: *Proceedings of 2015 Forth International Conference on e-Technologies and Networks for Development* (ICeND), Lodz, Poland, 2015, pp. 1 – 4, DOI: 10.1109/ICeND.2015.7328539.

[9] S. Nayak, M. K. Gourisaria, M. Pandey and S. S. Rautaray, Prediction of heart disease by mining frequent items and classification techniques, in: *Proceedings of 2019 International Conference on Intelligent Computing and Control Systems* (ICCS), Madurai, India, 2019, pp. 607 – 611, DOI: 10.1109/ICCS45141.2019.9065805.

[10] P. R. Puneeth and K. P. Rao, A comparative study on Apriori and reverse apriori in generation of frequent item set, in: *Proceedings of 2019 1st International Conference on Advances in Information Technology* (ICAIT), Chikmagalur, India, 2019, pp. 337 – 341, DOI: 10.1109/ICAIT47043.2019.8987430.

[11] K. Rajeswari and V. Vaithiyanathan, A novel method for frequent pattern mining, *International Journal of Engineering and Technology* **5**(3) (2013), 2150 – 2154, URL: https://www.enggjournals.com/ijet/docs/IJET13-05-03-031.pdf.

[12] V. Vaithiyanathan, K. Rajeswari, R. Phalnikar and S. Tonge, Improved apriori algorithm based on selection criterion, in: *Proceedings of 2012 IEEE International Conference on Computational Intelligence and Computing Research*, Coimbatore, India, 2012, pp. 1 – 4, DOI: 10.1109/ICCIC.2012.6510229.

[13] C. Yadav, S. Wang and M. Kumar, An approach to improve apriori algorithm based on association rule mining, in: *Proceedings of 2013 Fourth International Conference on Computing, Communications and Networking Technologies* (ICCCNT), Tiruchengode, India, 2013, pp. 1 – 9, DOI: 10.1109/ICCCNT.2013.6726678.

[14] Q. Yang, Q. Fu, C. Wang and J. Yang, A matrix-based Apriori algorithm improvement, in: *Proceedings of 2018 IEEE Third International Conference on Data Science in Cyberspace* (DSC), Guangzhou, China, 2018, pp. 824 – 828, DOI: 10.1109/DSC.2018.00132.

[15] J. Yang, Z. Li, W. Xiang and L. Xiao, An improved Apriori algorithm based on features, in: *Proceedings of 2013 Ninth International Conference on Computational Intelligence and Security*, Emeishan, China, 2013, pp. 125 – 128, DOI: 10.1109/CIS.2013.33.