Advances in Computer, Signals and Systems (2023)
Clausius Scientific Press, Canada

DOI: 10.23977/acss.2023.070810
ISSN 2371-8838 Vol. 7 Num. 8

An Approach of Improved Traversal Merging of Transaction Data for Faster Apriori Algorithm

Xubo Wu[1], Huan Fang[1,a,*], Xiangyu Zhang[1]

[1]School of Mathematics and Big Data, Anhui University of Science and Technology, Huainan, Anhui, China
[a]fanghuan0307@163.com
[*]Correspondence author

Keywords: Apriori algorithm, fast traversal merging method, adaptive threshold determination, central limit theorem, shopping blue algorithm

Abstract: In order to improve the operational efficiency of the Apriori algorithm in the data preprocessing stage of large-scale data and achieve overall optimization of the Apriori project, a fast traversal merge pre-processing method is proposed by integrating an adaptive association mining threshold determination method. Firstly, the proposed fast traversal merging method is analyzed and compared with two benchmark algorithms, and the experimental results show that the running time of the fast traversal merging method is much lower than that of the two benchmark methods; secondly, according to the central limit theorem, a data adaptive support threshold setting method is proposed, which can avoid the subjectivity of the minimum support threshold setting in association mining; finally, the two proposed algorithms are applied to Apriori and the results show that the application of the proposed improved method for association mining gives significantly better results than association mining under the better processing of the benchmark algorithm, and thus can significantly improve the efficiency of solving the shopping basket problem.

1. Introduction

In today's increasingly competitive retail world, it has become a major concern for retailers to find the best product mix and match, to understand consumers' shopping habits and to develop marketing strategies, which helps to maximize the benefits. The Apriori algorithm is a common used algorithm for mining association rules, and it can discover the association relationships between different products in shopping basket data. Therefore, the Apriori algorithm can be designed to investigate the purchasing behavior of customers, which supports decision-making of retailers to develop marketing strategies. However, the processed data is restricted by specific requirements, and most transaction data exported from databases doesn’t meet the requirements.

Hence, preparing the data in required formatting is a common thing before the Apriori algorithm application, and it is also a time-consuming stage, especially when the datasets are in large-scale size. In order to enhance the overall efficiency of the Apriori algorithm, this paper proposes an improved traversal merging of transaction data, which can realize faster data preparing. Generally,

the overall performance enhancement of the Apriori algorithm is realized from two perspectives: on the one hand, optimize the data preprocessing stage to improve the efficiency of generating a "shopping basket" from a large amount of transaction data directly exported from the database; on the other hand, the Apriori algorithm itself has been optimized to improve the performance of association rule mining.

This paper aims to improve the overall efficiency of the Apriori algorithm, from the perspective of combining both the data preprocessing and the Apriori algorithm itself together. Conclusively, there are two main contributions of this paper, which are shown as the follows on.

(1) Constructing an approach of fast traversal merging of transaction data. Under the practical application scenario of the Apriori algorithm, it is common that the direct use of transaction data exported from databases will inevitably increase the execution time. For the reason that, the exported transaction data expressed by data list should be merged in the proper required formatting.

(2) Proposing a self-adaptive thresholds determination method for the Apriori algorithm. Its main highlight lies in the fact that the self-adaptive thresholds can be adjusted by the real data distribution, and thus complete the target of the more reasonable and interpretable mining results.

In the following parts, a brief literature review is introduced in section 2, and the proposed method of fast traversal merging is proposed in section 3. A series of experiments are conducted for validate the effectiveness of the proposed method, and an in-depth analysis is presented in section 4, including comparisons with some related baseline methods. Furthermore, a self-adaptive thresholds determination algorithm is introduced in section 5, and an applicable case study is designed to test the generality, feasibility and effectiveness of our proposed method in section 6. Finally, conclusions and future work are presented in section 7.

2. Related work

2.1 Application of the Apriori algorithm

In addressing the improvement of the Apriori algorithm, various related research findings have emerged.[1] introduces a hybrid optimization approach using differential evolution and the sine cosine algorithm to automatically adjust numerical attribute intervals and mine numerical association rules. This method boasts strong adaptability and automation.[2] proposes a novel representation scheme based on chaos numbers in evolutionary computation for quantitative association rule mining.[3] presents the Butterfly Optimization Algorithm (BOA) to enhance the efficiency of basic algorithms during association rule mining, employing both CPU and GPU parallelization for synchronization and rule exploration operations.[4] introduces a new data analysis method called Apriori probabilistic analysis, focusing on scalability and using compressed bitmap structures to reduce memory usage and computation time.[5] enhances the Apriori algorithm by avoiding redundant database scans through the use of a two-dimensional array, reducing execution time and improving the effectiveness of frequent item set mining.Additional studies [6-8] demonstrate the general applicability and feasibility of the Apriori algorithm in various association rule mining scenarios.

2.2 Parameters setting method of Apriori algorithm

Determining appropriate values for min_support and min_confidence is crucial in association rule mining but often challenging. Currently, these values are typically set based on industry characteristics or domain experts' experience. For example, [6] utilized the Apriori algorithm on personal credit data in commercial banks, setting min_support = 10% and min_confidence = 90% after extensive analysis. This configuration yielded superior performance compared to other

settings.Similarly, in [7], the Apriori algorithm's association rules were employed for equipment warehouse cargo space allocation, with min_support = 0.26 and min_confidence = 0.8 chosen through parameter setting comparisons. In [8], min_support and min_confidence were set to 0.2 and 0.6, respectively, based on expert suggestions and experimental analysis.

A concise literature review reveals that min_support values lack regularity, often determined through exhaustive methods, which are laborious and subjective. This paper proposes a data adaptation approach using the central limit theorem to adaptively derive min_support values from the data itself. The central limit theorem, ensuring the sample mean distribution approximates normal with a sufficiently large sample size, serves as the theoretical foundation for this adaptive approach.

2.3 Summary of related work

As mentioned above, it can be deduced that most of the state-of-the-art methods only focus on optimizing the Apriori algorithm itself, but rarely consider optimizing its corresponding data pre-processing stage, which can also improve the overall efficiency of the Apriori algorithm. From this point, this paper aims to improve the overall efficiency of the Apriori algorithm, from the perspective of combining both the data preprocessing with the Apriori algorithm itself together.

3. Proposed method of fast traversal merging for shopping basket data

3.1 Characteristics of transaction data to be processed

To summarize, this paper begins by describing the data used, which comprises supermarket sales list data with specific attributes. (Table 1)

(1) Order numbers for purchase identification.

(2) Date and timestamp of sales records.

(3) Product names, among others.

Table 1: Selected supermarket sales list data

Id	Date	Name
0420022301010001	2023-01-01 07:24:55	"Moco Rondo" High Calcium Cheese Sticks
0220022301010001	2023-01-01 07:49:01	Amushi Flavoured Yogurt Mango Oatmeal
0420022301010002	2023-01-01 07:52:56	Shuanghui 240G King of Kings
0420022301010002	2023-01-01 07:52:56	Kangshifu Super Cooler
0420022301010003	2023-01-01 08:05:50	190G Black Tea Toothpaste
0420022301010003	2023-01-01 08:05:50	One Brush Premium Kids
0420022301010003	2023-01-01 08:05:50	Kiss Clean 212 Toothbrush
0420022301010003	2023-01-01 08:05:50	All Purpose Soap Powder
0420022301010004	2023-01-01 08:07:03	Miyo Kids Toothbrush 920
0420022301010004	2023-01-01 08:07:03	One Brush Premium Kids
0420022301010004	2023-01-01 08:07:03	Mouthwash Apple Propolis Children's Toothpaste

The data format involves each order number corresponding to one or more products, resulting in consecutive rows for the same order number, making it necessary to transform this format into the required "shopping basket" format for Apriori. This conversion presents challenges with existing methods. To address this data preprocessing issue efficiently, the paper introduces a fast traversal algorithm. This algorithm leverages the format characteristics of data with the same order number in the supermarket sales list, minimizing unnecessary traversal and ensuring each data traversal progresses towards the desired format for the Apriori algorithm. (Figure 1)

id	baskets
420022301010001	"Moco Rondo" High Calcium Cheese Sticks
420022301010002	Shuanghui 240G King of Kings,Kangshifu Super Cooler
420022301010003	190G Black Tea Toothpaste,One Brush Premium Kids,Kiss Clean 212 Toothbrush,All Purpose Soap Powder
420022301010004	Miyo Kids Toothbrush 920,One Brush Premium Kids,Mouthwash Apple Propolis Children's Toothpaste

Figure 1: Consolidated results for selected supermarket sales list data

3.2 Designed fast merging algorithm framework

The proposed fast traversal merging algorithm flow chart is shown in Fig. 2, where *data_id* denotes the order number collection, *data* denotes the inventory dataset, *temp* denotes the intermediate variable for staging the matched data, *baskets* stores the target dataset, *i* and *j* denote related pointer variables, and *n* denotes the count variable.

The procedures of the data processing are demonstrated as the follows on. Firstly, when *j* is less than the length of the dataset data, traverse the *data_id* find the first data in data that matches the order number of *data_id*[*i*]; then, traverse down data and add the data to the temp list until there is no matching data,and then assign the number of data and match *data_id*[*i*] in *temp* to *n;*next change the style of the data in *temp* according to the required style and add it to *baskets*, and Exiting the innermost loop. Furthermore, exit the sub inner loop and set j and i to j=j+n, i=i+1. Finally, continue the loop from the beginning until *i* is no longer less than the length of the *data_id* dataset.

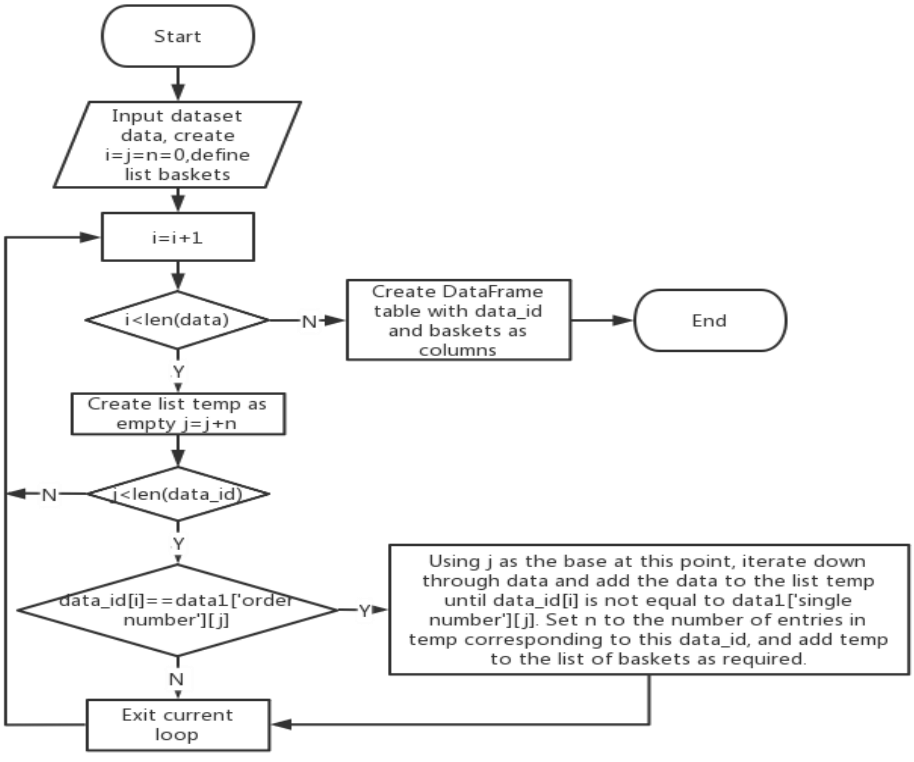

Figure 2: Flow chart of the proposed fast traversal merging algorithm

Algorithm 1: Fast traversal merging algorithm for transaction data.
Input: order number set *data_id*, list data set *data*.
Output: target dataset *baskets*.

```
1: j=0, baskets=[]
2: for i in range(len(data_id)):
3:         temp=[]
4:         while j<len(data['single number']):
5:             if(data_id[i]==data['single number'][j]):
6:                 a=j, b=j+100
7:                 for x in range(a,b):
8:                     if(x<len(data) and data_id[i]==data['single number'][x]):
9:                         temp.append(data['name'][x])
10:                    else:
11:                        n=len(temp), tem3=','.join(temp), baskets.append(tem3)
12:                        break
13:            break
14:        j=j+n
15:return baskets
```

3.3 Pseudo codes

The fast traversal merge algorithm proposed in this article is implemented in Python language, a nd pseudo code is described in Algorithm 1.

The aim of lines 4 and 5 in Algorithm 1 is to find out where in data the data_id with subscript i matches for the first time. The purpose of line 6 is to set the starting point and the ending point for the lookup of the current order number, noting that the ending point b is not necessarily reachable, where 100 is user-defined and this value only needs to be greater than the maximum value of the item corresponding to the order number in this data.

4. Experimental analysis and comparisons

4.1 Experimental setting

Firstly, for the sake of simplifying, some basic notations are introduced here. We use *Gi* to denote the first group of data, and there exist 500 list data in group *G*1, 1000 list data for *G*2 ..... and so on, up to the 20th group *G*20, where group *i* has 500 more list data than group *i*-1.

Secondly, two kinds of algorithms are taken as baselines to compare with the proposed fast traversal merging algorithm in this paper. The algorithms to be compared are listed in Table 2.

Table 2: Three algorithms of data merging required by the Apriori algorithm.

Algorithm name	Literature	Time complexity
Traversal algorithm	[9]	O(*n***m*)
Fold-and-half algorithm	[9]	O($\log_2 n$!)
Fast traversal merge algorithm	This paper	O(*n*)

Thirdly, 10-round tests are conducted on each of the list three algorithms on the basis of the same set of data, and the execution times are obtained as the average of the run times (in seconds) of the 10-round tests for each algorithm, which is used as the experimental results of the algorithms. The final 20-round test results corresponding to each algorithm are grouped under their respective data lists, and the test curves under the three algorithms are depicted for visualizing the experimental

results.

4.2 Comparison with the existing methods

In this section, two baseline algorithms are introduced: the traversal algorithm and the half-folding algorithm [9]. The traversal algorithm involves indiscriminating data traversal, leading to numerous repetitive and meaningless traversals. Conversely, the half-folding algorithm, while reducing some meaningless traversals, restarts each retrieval from scratch. To overcome these limitations, the proposed fast traversal algorithm capitalizes on formatting characteristics, specifically the presence of transaction data with the same order number in the supermarket sales list. This approach minimizes unnecessary traversals, allowing each traversal to progress more efficiently toward the desired format.

The traversal algorithm directly iterates by searching for data with a given single number, while the half-folding algorithm employs a classic half-folding strategy on the list after eliminating matching data. The proposed fast traversal merging algorithm in this paper starts by identifying the sequence number of the first match to a given single number. It then initiates a downward traversal from this point, continuing until no further matches are found. The number of matched items is added to the sequence number of the first match, and the process iterates for the next tracking number until all tracking numbers have been matched. This algorithm is illustrated in Algorithm 1.

4.2.1 Theoretical time complexity comparisons

Assuming that there be m single numbers, n lists, with X_i (i is a positive integer) is the number of lists corresponding to the i-th single number, the corresponding time complexity of each algorithm is analyzed as the follows on.

(1) As for the traversal algorithm T_1, the time complexity T_1 is $O(T_1) = O(m*n)$;

(2) As for the fold-and-half algorithm T_2, in this article, in general, the number of executions is shown as in eq.(1).

$$\sum_{i=1}^{i=m}\sum_{j=1}^{j=X_i}\log_2(n-\sum_{k=1}^{k=i-1}X_k-j+1) \quad \text{(where } i, j, k=1,2,3,4,5..., \text{ which are positive integers.)} \tag{1}$$

When $m=n$, each order number corresponds to only one list data, i.e. $Xi=1$, or when $m=1$, one order number corresponds to all lists, i.e. $Xi=n$. At this point, the number of executions both can be expressed as eq.(2) .

$$\sum_{i=1}^{i=m}\log_2(n-i+1) \quad \text{(where } i=1,2,3,4,5... \text{ , which is a positive integer)} \tag{2}$$

And the time complexity $O(T2)=O(\log_2 n!)$ can be calculated.In order to facilitate the subsequent comparison of time complexity, the scaling method is used to scale the equation (2) to between $(n, n\log_2 n)$, i.e. the range of time complexity is $O(n)<O(T2)<O(n\log_2 n)$. It should be noted that $O(T2)$ is closer to $O(n\log_2 n)$. In addition to the above situation, we set Xi to any value and satisfy $X1+X2+X3+X4+X5...+Xm=n$. It can be seen that no matter how much Xi is equal to and it must go through eq.(3).

$$\sum_{j=1}^{j=X_i}\log_2(n-\sum_{k=1}^{k=i-1}X_k-j+1) \text{ (where } i, j, k=1,2,3,4,5..., \text{ which are positive integers)} \tag{3}$$

Times to fully search for the order number corresponding to Xi in the supermarket list, and the time complexity $O(T2)$ remains $O(\log_2 n!)$. In summary, no matter what value Xi takes, the total

number of executions will not change, that is, the time complexity O ($T2$)=O($\log_2 n!$), and O (n)<O ($T2$)<O ($n\log_2 n$).

(3)As for the proposed algorithm, i.e., fast traversal merging algorithm T_3, the time complexity of is calculated as O (T_3)=O(n).

In summary, for the sake of brevity, the theoretical time complexity of the three algorithms is shown in Table 2. It is obvious that the time complexity of the three algorithms is sorted as O (T_3)<O(T_2)<O(T_1), which means that the proposed fast traversal merging algorithm is superior to the mentioned baseline algorithms.

4.2.2 Time execution time comparisons

The ten parallel tests were performed on each of the 20 sets of transaction data based on each algorithm, and the average of the running times (in seconds) of the 10 parallel tests performed on each algorithm was taken as the test result for that set of data on that algorithm, and the experimental results were shown in Tables 3 and 4, with a visual graph comparison shown in Fig. 3. The experimental results show that, of the three algorithms, the proposed fast traversal merge algorithm consistently has a lower execution time than the other two algorithms, which is significantly different from the other two algorithms, and this difference becomes more pronounced as the amount of data increasing.

Table 3: Running time of each algorithm on different data volumes

Data size / Algorithm	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
Fast traversal merge algorithm	0.0	0.0	0.0	0.01	0.01	0.01	0.01	0.01	0.01	0.01
Fold-and-half algorithm	0.4	0.81	1.27	1.72	2.2	2.71	3.18	3.74	4.3	4.82
Traversal algorithm	0.36	1.41	3.32	6.07	9.23	13.65	18.45	24.45	31.14	38.74

Table 4: Running time of each algorithm on different data volumes

Data size / Algorithm	5500	6000	6500	7000	7500	8000	8500	9000	9500	10000
Fast traversal merge algorithm	0.01	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.03
Fold-and-half algorithm	5.37	6.03	6.56	7.14	7.97	8.58	9.29	9.88	10.69	11.38
Traversal algorithm	46.94	56.2	66.06	77.1	88.36	100.34	113.19	125.83	139.73	155.49

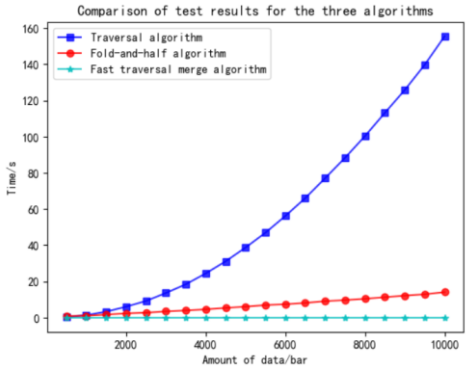

Figure 3: Running time comparison of the three algorithms on different data volumes

4.2.3 Effectiveness comparisons

The Friedman test is a non-parametric test used to compare multiple sets of data to determine whether they are significantly different. If the *p-value* of the Friedman test is less than the significance level (usually 0.05), the hypothesis is rejected and the data are considered as significantly different. If the Friedman test is significant, a subsequent multiple comparison analysis can be performed, such as a rank operation on each group of data, calculating the average rank value of each group of data, and then determine which algorithm performs the bast efficient based on the average rank value. In this paper, the above scheme is used to verify whether there is a significant difference between the three algorithms, and to give a rank of the three algorithms in terms of execution efficiency if there is a significant difference. The results are summarised in Table 5.

Table 5: Results of non-parametric tests and multiple comparison analysis

Algorithm name	P_value	Average_rank
Traversal algorithm	5.329544830 873161e-09	3
Fold-and-half algorithm		2
Fast traversal merge algorithm		1

In Table 5, we can see that *P_ Value*=5.329544830873161e-09, which is very small and far less than the significance level of 0.05. This indicates that our results are very significant, indicating that there are indeed significant differences in the performance of the three algorithms. Secondly, based on Average_ Rank can know that the Fast traversal merge algorithm has the highest execution efficiency among the three algorithms.

In summary, there exist significant differences among the mentioned three algorithms, and it is clear that the proposed fast traversal merging method is significantly better than the other two methods, especially when the data is of large size.

5. Self-adaptive threshold determination of the Apriori algorithm

This section focuses on parameter setting for min_support and presents a self-adaptive threshold determination algorithm. To achieve a reasonable min_support, several considerations are made.

Firstly, the generation of strong association rules relies on frequent item sets, which are sets of items occurring together frequently in the data. To ensure reliable and effective strong association rules, it's important to retain a moderate number of frequent items. Secondly, by arranging item frequencies in ascending order and using the mean value, a balanced threshold can be established. The mean is selected as the min_support value, ensuring that the number of frequent items remains moderate, striking a balance between strictness and looseness. Lastly, the central limit theorem is employed to estimate the mean of the entire sample. This theorem, a fundamental concept in probability theory, explains that the mean of a large number of independent and identically distributed random variables tends to follow a normal distribution under certain conditions.

The analysis of supermarket sales data reveals that many items have extreme low sales volumes. To reduce interference from such items, a data adaptation method involving random sampling and recursive standard deviation calculations is employed. The stability of standard deviation values over multiple iterations indicates the suitability of using the mean as a substitute for min_support.Ultimately, the central limit theorem is applied to estimate the sample mean, yielding a min_support= 0.0019 (with 4 decimal places). (Table 6 and 7)

Algorithm 2:Adaptive threshold determination algorithm for min_support.
Input: supermarket sales data.
Output: min_support.

1: A random sample of 1000 items (the larger the sample size the better) is taken from the sales records, combined and the frequency of each item is calculated and the frequencies are considered as a new sample set.
2: Calculate the mean of the sample set.
3: Repeat steps 1 and 2 for 50 times to obtain 50 sample means.
4: Calculate the mean of the 50 sample means, which is the estimated mean of the original sample commodity frequency.
5: Let *min_support* be equal to the mean estimate of the frequency of the commodity in the original sample
6:return *min_support*

Table 6: Summary of merchandise sales volume statistics

Statistical scope	Sales equal to 1	Sales less than 6
Quantity/type	776	2372

Table 7: Summary of standard deviation results

Recursive times	1	2	3	4	5	6	7	8	9	10
Standard deviation	4.72	4.13	3.69	3.45	3.23	3.03	2.86	2.71	2.58	2.47

6. Case study

6.1 Applicable scenario

The Apriori algorithm is a common algorithm for mining association rules. It can be used to achieve shopping basket analysis, to discover the association between different products in supermarkets, to study the purchasing behaviour of customers, and to assist retail companies in formulating marketing strategies.

6.2 Association rule mining

In this paper, we use the Python language to implement the Apriori algorithm, utilize the data prepocessing results obtained by the proposed fast traversal merging algorithm, and use the parameters determined by self-adaptive algorithm. The final experimental results are listed as the examples shown in Fig. 4.

antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
(Flammulina velutipes)	(Sanquan Meatballs)	0.016192	0.014294	0.004076	0.251724	17.610857	0.003844	1.317303
(Sanquan Meatballs)	(Flammulina velutipes)	0.014294	0.016192	0.004076	0.285156	17.610857	0.003844	1.376256
(Pork Blood)	(Tofu)	0.005137	0.041374	0.003071	0.597826	14.449481	0.002858	2.383612
(Tofu)	(Pork Blood)	0.041374	0.005137	0.003071	0.074224	14.449481	0.002858	1.074626
(Garlic)	(Ginger)	0.010609	0.022781	0.003015	0.284211	12.476006	0.002773	1.365233
(Ginger)	(Garlic)	0.022781	0.010609	0.003015	0.132353	12.476006	0.002773	1.140316
(Sanquan Meatballs)	(Lettuce)	0.014294	0.022446	0.003797	0.265625	11.834188	0.003476	1.331138
(Lettuce)	(Sanquan Meatballs)	0.022446	0.014294	0.003797	0.169154	11.834188	0.003476	1.186389

Figure 4: Examples of association rule mining results.

6.3 Analysis of association mining results

From Fig. 4, it is shown that the support for "Flammulina velutipes" and "Sanquan Meatballs" and "Lettuce" and "Sanquan Meatballs" is comparative high. The support level for these two groups of products is high, which means that customers are more likely to buy these two groups of products at the same time. In addition, 25.2% of customers bought "Sanquan meatballs" at the same

time as "Flammulina velutipes". After purchasing "Sanquan Meatballs", 28.5% of customers also purchased "Flammulina velutipes". The lift between the two products was 17.61, which is significantly higher than 1. Therefore, the two products are positively correlated and have a mutual promotion effect. Therefore, supermarkets can consider these types of product when placing and stocking goods, and when running promotions, supermarkets can consider selling "Flammulina velutipes", "Lettuce" and "Sanquan Meatballs" packaged. Alternatively, when run an event, sell one of the three at a discount to increase sales of the other two items.

7. Conclusions and future work

This paper proposes a pre-processing method for scenarios in which the Apriori algorithm processes supermarket sales list data. Through a fast traversal merging method, the data pre-processing stage is successfully optimized, thus further improving the efficiency and accuracy of the overall Apriori algorithm. The proposed method not only avoids the problem of algorithmic time and space complexity due to the large scale of data, but also improves the efficiency and accuracy of association rule mining, which helps retail enterprises to better study customers' purchasing behavior and develop marketing strategies to maximize benefits. In addition, this paper uses a self-adaptive parameters determination strategy using central limit theorem to improve the efficiency and accuracy of the algorithm. The experimental results show that our proposed method is faster than typical baseline method statically.

In the future, other research directions can be explored, including applying other self-adaptive parameters setting method to enhance the performance of association rule mining.

References

[1] E. V. Altay, A. Bilal, "Chaos numbers based a new representation scheme for evolutionary computation: Applications in evolutionary association rule mining", Concurrency and Computation: Practice and Experience, vol. 34, no.5, 2022, pp. e6744.

[2] P. He, B. Zhang, and S. Shen, "Effects of out-of-hospital continuous nursing on postoperative breast cancer patients by medical big data," Journal of Healthcare Engineering, vol. 2022, 14 pages, 2022.

[3] D. Suo, Z. Zhang, "Parallel design of apriori algorithm based on the method of "determine infrequent items & remove infrequent itemsets" ", IOP conference series: earth and environmental science, vol. 634. no. 1. IOP Publishing, 2021.

[4] A. Colombo et al. "Apriori-roaring: frequent pattern mining method based on compressed bitmaps", International Journal of Business Intelligence and Data Mining, vol. 21, no. 1, 2022, pp. 48-65.

[5] D. Liu, "Construction of Higher Education Management and Student Achievement Evaluation Mechanism Based on Apriori Algorithm", Mobile Information Systems, vol. 2022, 9 pages, 2022.

[6] Z. Lin, "Application of association rules in personal credit audit of commercial banks--an analysis of the application based on Apriori algorithm", Business Accounting, vol. 10, 2022, pp. 60-63.

[7] Y. Lv, X. Dong, F. Wang, C. Ren, "Application of association rules based on Apriori algorithm in equipment warehouse cargo space allocation", Journal of the College of Ordnance Engineering, vol. 28, no. 5, 2016, pp. 38-42.

[8] X. Zhao, H. Huo, S. Pang, "Identification of Environmental Pollutants in Construction Site Monitoring Using Association Rule Mining and Ontology-Based Reasoning", Buildings, vol. 12, no. 12, 2022, pp. 2111.

[9] W. Yan, W. Wu, Data structure, Beijing: Tsinghua University Press, 2018.