VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
UNIVERSITY OF INFORMATION TECHNOLOGY
FACULTY OF COMPUTER SCIENCE

FINAL REPORT
APRIORI ALGORITHM

Instructor

Ph.D. Tiep Vinh Nguyen

Students

Thang Dinh Duong - 19522195
Nhat Minh Phan - 19521956
Tri Dinh Duc Truong - 19522395

Class

CS116.M12.KHCL

Ho Chi Minh City - 2021

Contents

1 Introduction

When you go to a supermarket, you may find bread, butter, and jam bundled together. Because the seller knows that people who buy bread also buy butter or jam, so they bunching them together to make it comfortable for the customer to buy these items in the same place. At the same time, it's also increases the sales performance. This is called Market Basket Analysis.

Market Basket Analysis works based on Association Rule Mining. There are many methods to perform Association Rule Mining. The Apriori algorithm that we are going to introduce in this report is the most simple and straightforward approach.

Figure 1: Association Rule in Market Basket Analysis

In this report, we introduce the association rule mining problem and the Apriori algorithm to tackle this challenge. We will have an insight about what the Apriori algorithm is; why it can be used for association rule mining and how it is executed step-by-step. We also apply this technique on some dataset to experiment the performance, hyperparameter tuning for Apriori algorithm and have a comparison with other association rule mining algorithms.

2 Methods

2.1 Association Rule Mining

Before finding out what Association Rule Mining, we should understand some subproblem.

A set of items together is called an itemset. If any itemset has k items, it is called a k-itemset. An itemset that occurs frequently is called a frequent itemset. Therefore frequent itemset mining is a data mining technique to identify the items that often occur together.

The frequent itemset must satisfies a minimum threshold value for support. In addition, for frequent itemset mining method, it consider only those trans-

actions which meet minimum threshold support requirements.

Now we come to the main problem of this section, Association Rule Mining is identify the association between different variables in a database by searching frequent items in the dataset. One of the best and most popular examples of Association Rule Mining is the Market Basket Analysis. The problem analyses the association between various items that has the highest probability of being bought together by a customer[**2**].

Association Rule Mining is defined as: *"Let I= {...} be a set of 'n' binary attributes called items. Let D= {...} be set of transaction called database. Each transaction in D has a unique transaction ID and contains a subset of the items in I. A rule is defined as an implication of form X⇒Y where X,Y ? I and X?Y=?. The set of items X and Y are called antecedent and consequent of the rule respectively[**5**]."*

Association rule mining consists of 2 steps:

1. Find all the frequent itemsets.
2. Generate association rules from the above frequent itemsets.

Figure 2: Example for transactional dataset

Here is table of transaction for figure 2

	Pasta	TomatoSauce	RedWine	Seafood	WhiteWine	Salami
T1	x	x	x			
T2	x		x	x	x	x
T3	x	x	x			
T4				x	x	

With this set of transaction, we can find the rules that will predict the occurrence of an item based on the occurrences of other items in the transaction.

Before start generate association rules, let see some basic definitions.

- **Support Count(σ):** Frequency of occurrence of a itemset.
  For example: σ(Pasta, TomatoSauce, RedWine) = 2

- **Frequent Itemset:** An itemset whose support is greater than or equal to min_sup threshold (min_sup is decided by user).

- **Support(s):** The number of transactions that include items in the {X} and {Y} parts of the rule as a percentage of the total number of transaction.It is a measure of how frequently the collection of items occur together as a percentage of all transactions.

$$Support = \sigma(X \cup Y) \div Total \tag{1}$$

  It is interpreted as fraction of transactions that contain both X and Y.

- **Confidence(c):** It is the ratio of the no of transactions that includes all items in {Y} as well as the no of transactions that includes all items in {X} to the no of transactions that includes all items in {X}.

$$Confidence(X \Rightarrow Y) = Support(X \cup Y) \div Support(X) \tag{2}$$

  It measures how often each item in Y appears in transactions that contains items in X also.

- **Lift(l):** The lift of the rule X $\Rightarrow$ Y is the confidence of the rule divided by the expected confidence, assuming that the itemsets X and Y are independent of each other.The expected confidence is the confidence divided by the frequency of {Y}.

$$Lift(X \Rightarrow Y) = Confidence(X \Rightarrow Y) \div Support(Y) \tag{3}$$

  Lift value near 1 indicates X and Y almost often appear together as expected, greater than 1 means they appear together more than expected and less than 1 means they appear less than expected. Greater lift values indicate stronger association.

Example:

First, we must find all frequent itemsets, suppose the min_support = 0.5, our dataset has 4 transactions so all item set with Support Count greater than or equal to 2 (0.5 * 4 = 2) is frequent itemset such as: {Pasta}, {Pasta, RedWine}, {TomatoSauce, RedWine}, {Pasta, TomatoSauce, RedWine},...

Next we generate the association rule, take example with this rule: {Pasta, TomatoSauce} $\Rightarrow$ {RedWine}

s = σ(Pasta, TomatoSauce, RedWine) $\div$ Total = $2/4 = 0.5$

c = s $\div$ Sup(Pasta, TomatoSauce) = $0.5/0.5 = 1$

l = c $\div$ Sup(RedWine) = $1/0.75 = 1.33$

After this section, we had a overview knowledge about association rule, but how can we find out the frequent itemsets from the dataset ? The next section can answer this question.

2.2 Apriori algorithm

2.2.1 What is Apriori algorithm?

Figure 3: Apriori algorithm

Apriori algorithm was the first algorithm that was proposed for frequent itemset mining. It was later improved by R Agarwal [**11**] and R Srikant [**12**] in 1994 and came to be known as Apriori.

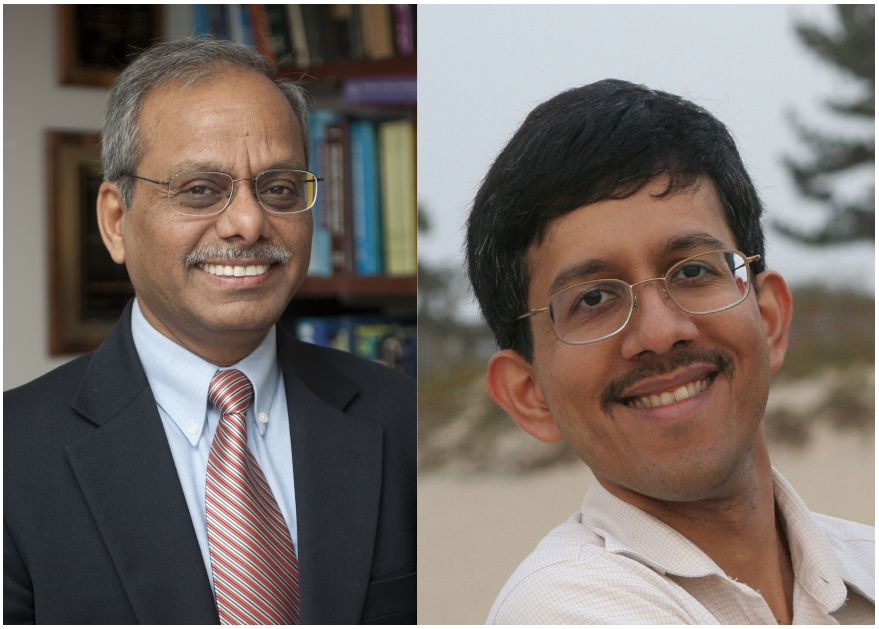

Figure 4: R Agarwal and R. Srikant

This algorithm attempts to operate on database records, particularly transactional records, or records including certain numbers of fields or items ***in order to finds the most frequent itemsets or elements in a transaction database and identifies association rules between the items*** just like the above-mentioned

example. Apriori uses a "bottom up" approach, where frequent subsets are extended one item at a time which is known as candidate generation, and groups of candidates are tested against the data.

Using breadth-first search and a Hash tree structure, apriori counts candidate item sets efficiently. It generates candidate item sets of length k from item sets of length k-1. Then it prunes the candidates which have an infrequent sub pattern. According to the downward closure lemma, the candidate set contains all frequent k-length item sets. After that, it scans the transaction database to determine frequent item sets among the candidates.

2.2.2 Why Apriori?

There are three popular algorithms of Association Rule Mining: Apriori (based on candidate generation), FP-Growth (based on without candidate generation) and Eclat (based on lattice traversal). But Apriori algorithm basically outperformed on almost all data sets by depth-first algorithms like Eclat or FP-growth [**9**]. Until now it's still one of the most basic and popular algorithms for association rules mining.

Apriori algorithm is considered as a brute-force method because this method considers every k-itemset as the candidate of frequent itemset [**8**]. Consider the computational needed for every candidate is O(k), as a whole algorithm, the complexity of this method is :

$$O(d2^{d-1})$$

Because of the simple process of this algorithm, the advantage of this algorithm is more comfortable to learn, understand, and implemented; this is the reason Apriori is called the most basic algorithm for association rules.

2.2.3 Apriori algorithm concept

Apriori algorithm is a sequence of steps to be followed to find the most frequent itemset in the given database. This data mining technique follows the join and the prune steps iteratively until the most frequent itemset is achieved. A minimum support threshold is given in the problem or it is assumed by the user.

Apriori says: The probability that item I is not frequent is if:

- P(I) $<$ minimum support threshold, then I is not frequent.
- P (I+A) $<$ minimum support threshold, then I+A is not frequent, where A also belongs to itemset.
- If an itemset set has value less than minimum support then all of its supersets will also fall below min support, and thus can be ignored. This property is called the Antimonotone property.

This algorithm uses two steps "join" and "prune" to reduce the search space. It is an iterative approach to discover the most frequent itemsets.

- **Join Step:** This step generates (K+1) itemset from K-itemsets by joining each item with itself.
- **Prune Step:** This step scans the count of each item in the database. If the candidate item does not meet minimum support, then it is regarded as infrequent and thus it is removed. This step is performed to reduce the size of the candidate itemsets.

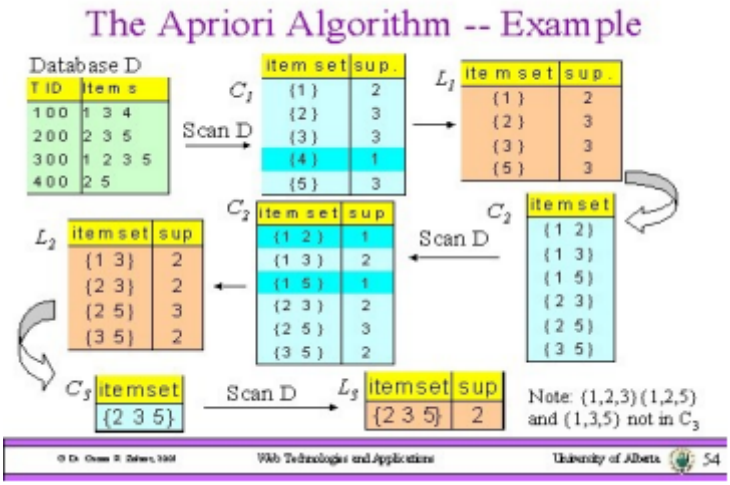

Figure 5: The Apriori Algorithm Example

2.2.4 Apriori Advantages and Disadvantages

- **Advantages of Apriori algorithm**
  - The Apriori algorithm is one of the most basic and popular algorithms for association rules mining.
  - This is the most simple and easy-to-understand algorithm among association rule learning algorithms.It's comfortable to learn, understand, and implemented.
  - The resulting rules are intuitive and easy to communicate to an end user.
  - It doesn't require labeled data as it is fully unsupervised; as a result, you can use it in many different situations because unlabeled data is often more accessible.
  - Many extensions were proposed for different use cases based on this implementation—for example, there are association learning algorithms that take into account the ordering of items, their number, and associated timestamps.

- The algorithm is exhaustive, so it finds all the rules with the specified support and confidence.
- It implements level-wise search.
- The join and prune steps of the algorithm can be easily implemented on large datasets.

- **Disadvantages of Apriori algorithm**
  - If the dataset is small, the algorithm can find many false associations that happened simply by chance. You can address this issue by evaluating obtained rules on the held-out test data for the support, confidence, lift, and conviction values.
  - It will loose mean if the maximum frequent can not be found fast.
  - It can not fit the situation that if there are still many items not included in the frequent set consisted of the maximum frequent itemsets and all of their nonempty subsets.
  - The overall performance can be reduced as it scans the database for multiple times.
  - The time complexity and space complexity of the Apriori algorithm is $O(2^{D-1})$, which is very high. Here D represents the horizontal width present in the database.
  - It needs more search space and computational cost is too high.
  - The time needed to hold a large number of candidate-sets with many frequent itemsets makes it slow.

2.3 Apriori's efficency improvement techniques

Due to some limitations of Apriori, many methods has been devised. Let see how they can improve this algorithm:

1. **Hash-Based Technique:** This method uses a hash-based structure called a hash table for generating the k-itemsets and its corresponding count. It uses a hash function for generating the table**[5]**. Specifically the 2-itemsets, since that is the way to enhancing execution. This calculation utilizes a hash based procedure to minimize the quantity of applicant item sets created in the 1st pass. It is guaranteed that the quantity of item sets in C2 produced utilizing hashing can be smalled, so that the output required to decide L2 is more efficient **[10]**.

2. **Transaction Reduction:** Due to large number of records in database results in much more I/O cost. Because of that me must reduces the size of the database. We create an attribute Size-Of-Transaction (SOT), containing number of items in individual transaction in database. The deletion process of transaction in database will made according to the value of K. Whatever the value of K, algorithm searches the same value for SOT in database. If value of K matches with value of SOT then delete only those transaction from database **[14]**.

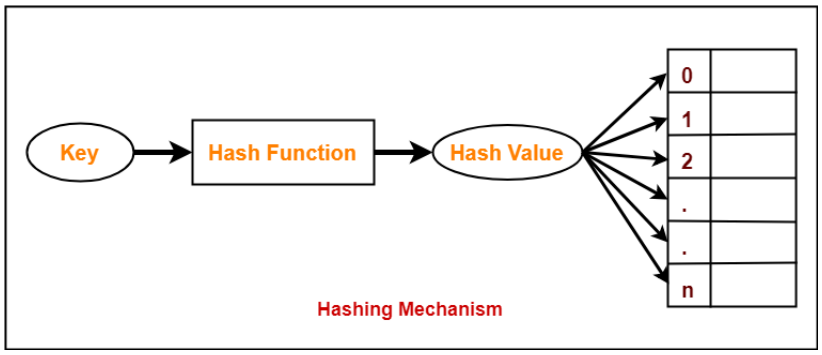

Figure 6: Illustration of Hash-Based technique

Figure 7: Illustration of Transaction Reduction

3. **Partitioning:** A partitioning technique can be used that needed two database scans to mine the frequent itemsets. It includes two phases involving In Phase I, the algorithm subdivides the transactions of D into n non-overlapping partitions. If the minimum support threshold for transactions in D is min_sup, therefore the minimum support count for a partition is min_sup × the number of transactions in that partition. For each partition, all frequent itemsets within the partition are discovered. These are defined as local frequent itemsets. The process employs a specific data structure that, for each itemset, records the TIDs of the transactions including the items in the itemset. This enables it to find all of the local frequent k-itemsets, for k = 1, 2... in only one scan of the database. A local frequent itemset can or cannot be frequently related to the whole database, D. Any itemset that is possibly frequent related D must appear as a frequent itemset is partially one of the partitions. Thus, all local frequent itemsets are candidate itemsets slightly D. The set of frequent itemsets from all partitions forms the worldwise candidate itemsets for D. In Phase II, the second scan of D is organized in which the actual support of each candidate is assessed to decide the global frequent itemsets [**13**].

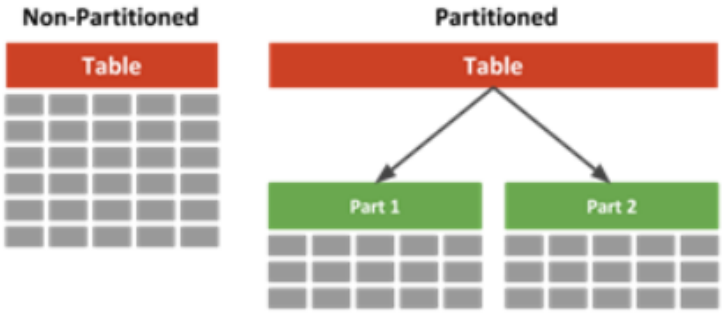

Figure 8: Illustration of Partitioning

4. **Sampling:** In 1996, Toivonen put forward Sampling algorithm The improvement of Apriori algorithm is Sampling algorithm, the main idea is that it using the sampling method to sample from the original database. In order to directly stored in memory, according to the sample database mining frequent itemsets, reduce the mining time. Sampling algorithm is using random sampling method to proceed with sampling, random sampling method has the characteristics of simple and quick Sampling algorithm put forward an important technology, as Negative Border. Negative Border's main idea is that according to the relevant model it get a threshold value which is smaller than minimum support. At the time of scanning the database to get frequent itemset F (i,min_sup) and support for itemset NB (i, Negative Border) between min_sup and Negative Border. Negative Border is applied in the extraction of sample can effectively avoid to frequent itemset loss.However, because data mining itself is decided to face the huge amounts of data, and dimension is also growing, in many cases, therefore, will produce a large number of candidate itemsets, in particular, 2-itemset, efficiency of mining is seriously affected. so, we have another way is to put forward SamplingHT algorithm what using the Hash table method to solve a large number of candidate 2-itemsets[**18**].

   (a) SamplingHT algorithm: The main idea is SamplingHT algorithm use Hash table technology to the traditional Sampling algorithm in the sample. The first time candidate 1-itemset and 2-itemset what after scanning the sample database can be obtained directly. Not only time that scan sample database is reduced, but also effectively reduce a large amount of infrequent candidate 2-itemset's generation, space and time is greatly reduced. After k-itemset also is such.

5. **Dynamic Itemset Counting:** Dynamic itemset technology, which can add the candidate itemset at different scanning point, divide the database into blocks from which to start a new scanning. It doesn't like the Apriori which fixes a new candidate after each complete database scanning. It can add new candidate at any start point. It dynamically values the support of all the counted itemsets. If all the subsets of an itemset are frequent, the itemset will be as a new candidate. Based on the definition

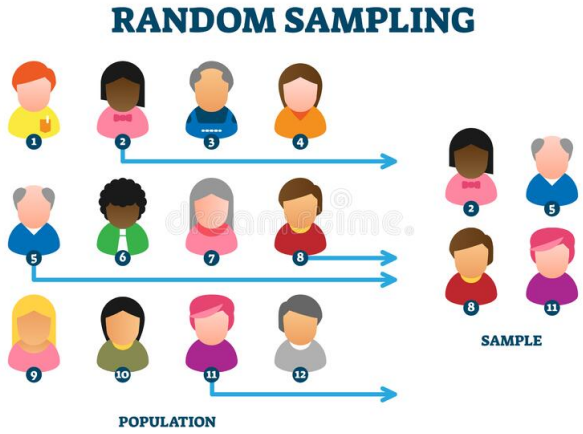

Figure 9: Illustration of Sampling

of the frequent itemset, if itemset I is not frequent (P(I) $<$ min_sup, if we add item A into I, the result itemsets (I $\cup$ A) can not be more frequent than I, so I $\cup$ A is not frequent.

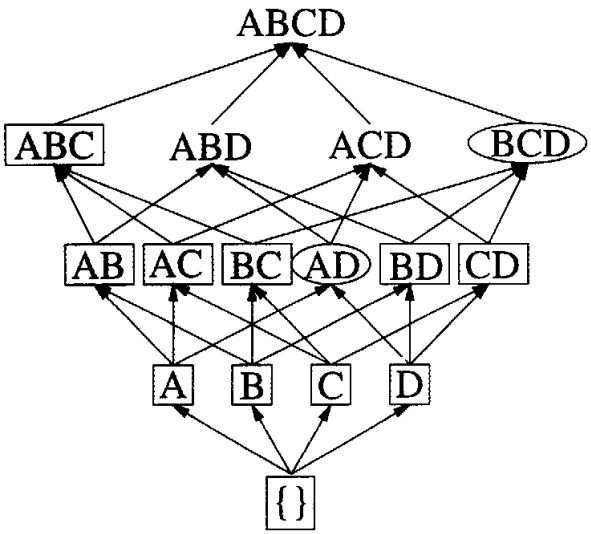

Figure 10: Illustration of Dynamic Itemset Counting

Base on this thought, an improved method by combining forward and reverse thinking; find the maximum frequent itemsets from the maximum itemset firstly, then get all the nonempty subsets of the frequent itemset. We know they are frequent on the basis of the Apriori's property. Secondly, scan the database once more from the lowest itemset and count the frequent. During this scanning, if one item is found out being excluded in the frequent set, it will be processed to judge weather the itemsets associated with it is frequent, if they are frequent, they will be

added in the barrel-strucure (include frequent itemsets). Thus, we get all the frequent itemsets. The key of this algorithm is to find the maximum frequent itemset fast. So it fits the situation that the dimensions of the maximum frequent and the maximum itemset's have a certain difference, but it is not too significant [**17**].

3 Experiments

3.1 Introduce about dataset

In this experiment, we apply Apriori to find frequent itemsets on two dataset, **store_data.csv** [**7**] and **grocery_dataset.csv** [**19**]. These are transactional dataset which each rows represent purchased items of a customer. Here is a quick glance of them:

	1	2	3	4	5	6	7
1	shrimp	almonds	avocado	vegetables mix	whole weat flour	yams	...
2	burgers	meatballs	eggs				
3	chutney						
4	...						

Table 1: The first three rows of **store_data.csv**

	1	2	3	4	5	6	7
1	circus fruit	semi-finished bread	margarine	ready soups			...
2	tropical fruit	yogurt	coffee				
3	whole milk						
4	...						

Table 2: The first three rows of **grocery_dataset.csv**

Below are a more detail information about these dataset, as **grocery_dataset.csv** is much more larger than **store_data.csv**, hence we only use **grocey_dataset.csv** as an additional speed test between Apriori and other algorithms:

Dataset	Rows	Columns	Items
store data	7501 rows.	Maximum is 20 columns (Which mean there are maximum 20 different items in one transaction).	119 unique items.
grocery dataset	9835 rows.	Maximum is 32 columns (Which mean there are maximum 32 different items in one transaction).	169 unique items.

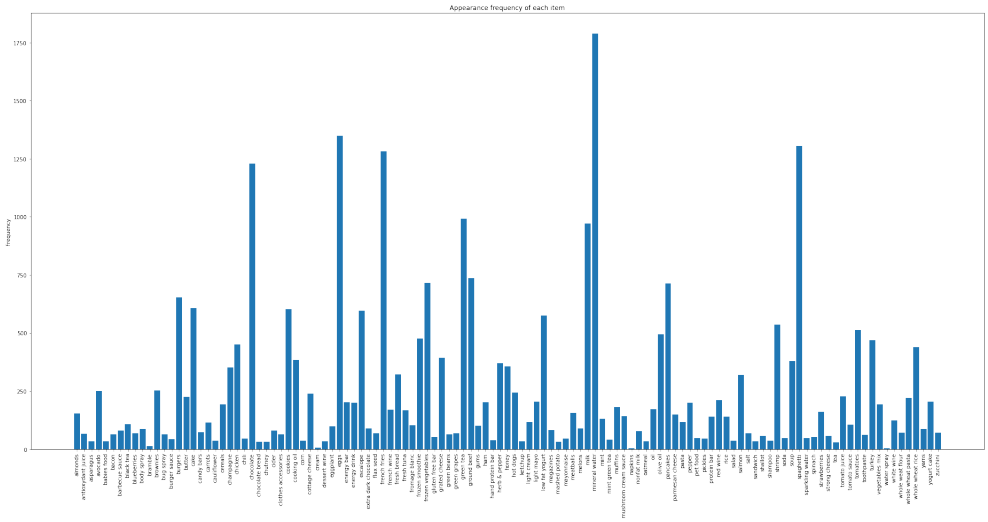

Figure 11: Histogram of **store_data.csv** dataset

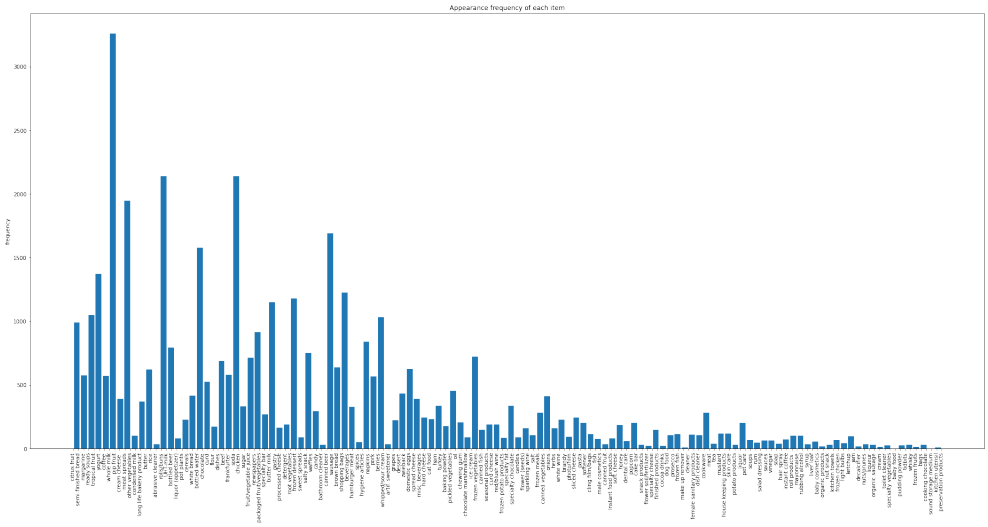

Figure 12: Histogram of **grocery_dataset.csv** dataset

3.2 Using Apriori algorithm in Python

We use Apriori algorithm from mlxtend library [15] and apply on **store_data.csv** dataset. First, we can find all frequent itemsets by:

```
from mlxtend.frequent_patterns import apriori
frequent_itemsets_apriori = apriori(
    df,
```

```
    min_support=0.002,
    use_colnames=True
)
```

This is the result of executing above code:

	support	itemsets
0	0.020397	(almonds)
1	0.008932	(antioxydant juice)
2	0.004666	(asparagus)
3	0.033329	(avocado)
4	0.004533	(babies food)
...	...	...
2430	0.002666	(spaghetti, milk, mineral water, turkey)
2431	0.002800	(spaghetti, olive oil, mineral water, pancakes)
2432	0.002266	(spaghetti, olive oil, mineral water, shrimp)
2433	0.002266	(spaghetti, olive oil, mineral water, tomatoes)
2434	0.002266	(spaghetti, soup, mineral water, pancakes)

Figure 13: Frequent itemsets founded by Apriori algorithm on **store_data.csv** dataset

Then we can generate association rules from frequent itemsets:

```
association_rules(
    frequent_itemsets_apriori,
    metric="confidence",
    min_threshold=0.7
)
```

We only consider the rules that have confidence ≥ 0.7, as showed in **Figure 14**. With this dataset, if we want to generate at least 1 association rule with confidence ≥ 0.7, min_support must be smaller than 0.003.

3.3 Determine the optimal hyper-parameters

The hyper-parameter of Apriori is min_support, the values of this parameter can be considered by user depending on their requirement. If min_suppport

	antecedents	consequents	support	confidence	lift
0	(mushroom cream sauce, pasta)	(escalope)	0.002533	0.950000	11.976387
1	(spaghetti, nonfat milk)	(mineral water)	0.002400	0.720000	3.020537
2	(red wine, turkey)	(mineral water)	0.002133	0.727273	3.051047
3	(chocolate, frozen vegetables, olive oil)	(mineral water)	0.002800	0.700000	2.936633
4	(spaghetti, eggs, cooking oil)	(mineral water)	0.002933	0.709677	2.977232
5	(spaghetti, pancakes, cooking oil)	(mineral water)	0.002133	0.727273	3.051047
6	(milk, olive oil, eggs)	(mineral water)	0.002666	0.714286	2.996564
7	(ground beef, shrimp, frozen vegetables)	(spaghetti)	0.002400	0.750000	4.307619
8	(soup, frozen vegetables, milk)	(mineral water)	0.003066	0.766667	3.216312
9	(frozen vegetables, olive oil, tomatoes)	(spaghetti)	0.002133	0.842105	4.836624
10	(spaghetti, soup, pancakes)	(mineral water)	0.002266	0.772727	3.241738

Figure 14: Association rules based on frequent itemsets found in **store_data.csv**

= 0.5, all found itemsets are occur together in at least 50% of all transactions in the dataset.

Deciding min_support is also depends on each dataset. Take an example with our dataset, this dataset is the list of transactions in one week, if we want to find all itemsets that are purchased at least 5 times on average a day, that mean 35 times a week, we can calculate min_support by get 35 divided by the total transactions: 35/7501 = 0.0045.

The smaller min_support the more itemset that satisfies the condition, which mean the more frequent itemset can be found.

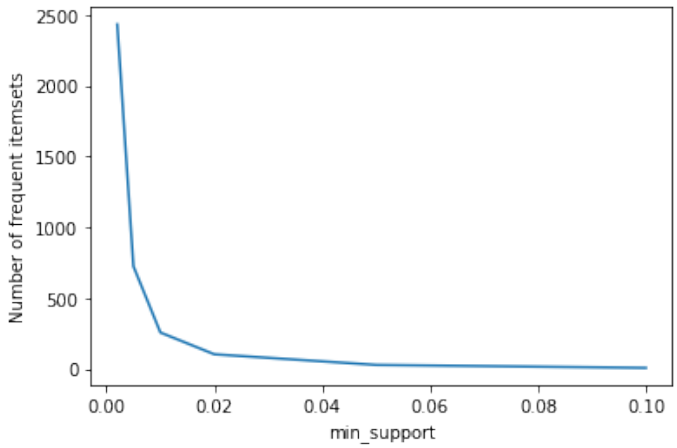

Figure 15: Number of frequent itemsets with different min_support

After a few times trying with different min_supports, we found that 0.238368

should be the maximum value of min_support for this dataset, that is, only one itemset could be archived.

	support	itemsets
0	0.238368	(mineral water)

Figure 16: Found itemset with maximum value of min_support

The found itemset has only one item is mineral water also is the item that appear the most in this dataset with 1788 times.

3.4 Compare Apriori with other algorithms

We compare Apriori's execution time with other frequent itemset mining algorithms, which includes FP-Max, FP-Growth. We evaluate execution time with values of min_support in range 0.002 to 0.01. As mentioned above, the smaller min_support the more frequent itemset can be found, which lead to the longer execution time.

Figure 17: Comparison the execution time of the three algorithms on **store_data.csv**

As showed in Figure 17 and Figure 18, we can observe that FP-Max and FP-Growth dominate Apriori at any stage of min_support, this somewhat describes the disadvantage in time execution of Apriori.

We also discover that if min_support = 0.238368 (maximum value for **store**

data dataset), Apriori executes in 28.8ms faster than FP-Max (68.8ms) and FP-Growth (60.7ms) but there are one frequent itemset can be found.

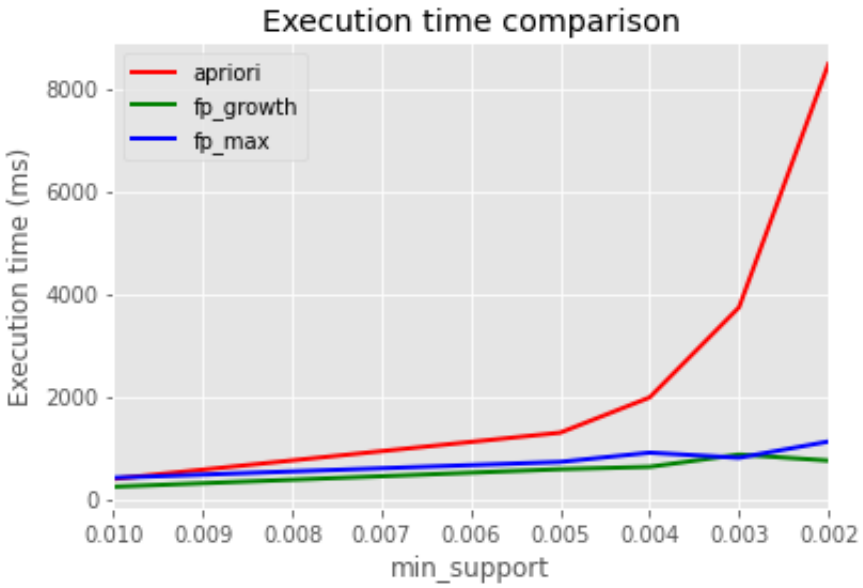

Figure 18: Comparison the execution time of the three algorithms on **grocery_dataset.csv**

4 Conclusion

In this document, we have introduced the Association Rule Mining problem, which involving in using the frequency and associations of itemsets in dataset to find the next relevant item in the set. Since real-world datasets frequently omit frequent itemsets, an algorithm to identify potentially relevant frequent itemsets is required, and Apriori is one of the finest algorithms to do so.

Beside the main concept of Apriori, we have also discussed about how to use this algorithm in Python and methods to improve it for better performance.

Finally, we have done some experiment with Apriori on a transactional dataset and also have a comparison between other Frequent Itemsets Mining algorithms.

References

[1] "Rule-based machine learning - Wikipedia", En.wikipedia.org, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Rule-

based_machine_learning.

[2] A. Nair, "Beginner's Guide To Understanding Apriori Algorithm With Implementation In Python", Analytics India Magazine, 2021. [Online]. Available: https://analyticsindiamag.com/beginners-guide-to-understanding-apriori-algorithm-with-implementation-in-python/.

[3] "Underrated Machine Learning Algorithms — APRIORI", Medium, 2021. [Online]. Available: https://towardsdatascience.com/underrated-machine-learning-algorithms-apriori-1b1d7a8b7bc.

[4] "Frequent Item set in Data set(Association Rule Mining) - GeeksforGeeks", GeeksforGeeks, 2021. [Online]. Available: https://www.geeksforgeeks.org/frequent-item-set-in-data-set-association-rule-mining/?fbclid=IwAR2-QISNDnxeRlBj_qMrMstY7ea9KRs0mENzsl9yMPnyDBbXfyCLsYpTdtg.

[5] "Apriori Algorithm in Data Mining: Implementation With Examples", Softwaretestinghelp.com, 2021. [Online]. Available: https://www.softwaretestinghelp.com/apriori-algorithm/.

[6] "Association Rule Mining - ML Wiki", Mlwiki.org, 2021. [Online]. Available: http://mlwiki.org/index.php/Association_Rule_Mining.

[7] "Association Rule Mining via Apriori Algorithm in Python," Stack Abuse, Aug. 09, 2018. https://stackabuse.com/association-rule-mining-via-apriori-algorithm-in-python/.

[8] WICAKSONO, D., JAMBAK, M. and SAPUTRA, D., 2019. The Comparison of Apriori Algorithm with Preprocessing and FP-Growth Algorithm for Finding Frequent Data Pattern in Association Rule. 1st ed. [ebook] Indonesia: Muhammad Ihsan JAMBAK, p.5. Available: https://www.atlantis-press.com/article/125939935.pdf

[9] Agrawal, R. and Srikant, R., n.d. Fast Algorithms for Mining Association Rules. 2nd ed. [ebook] 650 Harry Road, San Jose, CA 95120: Rakesh Agrawal, p.13. Available: http://www.vldb.org/conf/1994/P487.PDF

[10] J Doshi, A. and Joshi, B., 2018. Comparative analysis of Apriori and Apriori with hashing algorithm. 1st ed. [ebook] Department of Computer Engineering , Sardar Vallabhbhai patel Inst. of Tech, Vasad, India: Aesha J Doshi, Barkha Joshi, p.6. Available: https://www.irjet.net/archives/V5/i1/IRJET-V5I1206.pdf

[11] Agrawal, R., n.d. Rakesh Agrawal (computer scientist) - Wikipedia. [Online] En.wikipedia.org. Available: https://en.wikipedia.org/wiki/RakeshAgrawal(computerscientist)

[12] Srikant, R., n.d. Ramakrishnan Srikant - Wikipedia. [Online] En.wikipedia.org. Available: https://en.wikipedia.org/wiki/RamakrishnanSrikant

[13] Tutorialspoint.com. 2021. How can we further improve the efficiency of Apriori-based mining?. [Online] Available: https://www.tutorialspoint.com/how-can-we-further-improve-the-efficiency-of-apriori-based-mining

[14] Singh, J., Ram, H. and Sodhi, D., 2021. Improving Efficiency of Apriori Algorithm Using Transaction Reduction. 1st ed. [ebook] Department of Computer Science & Engineering, Amity School of Engineering and Technology, Amity University, Sec-125 NOIDA, (U.P.),India: International Journal of Scientific and Research Publications, p.6. Available: http://www.ijsrp.org/research-paper-1301/ijsrp-p1397.pdf

[15] S. Raschka, "Home - mlxtend", Rasbt.github.io, 2021. [Online]. Available: http://rasbt.github.io/mlxtend/.

[16] Sun, D., Teng, S., Zhang, W. and Zhu, H., 2007. An Algorithm to Improve the Effectiveness of Apriori. 2nd ed. [ebook] China: Haibin Zhu, p.7. Available: https://www.researchgate.net/publication/4278437-An-Algorithm-to-Improve-the-Effectiveness-of-Apriori/link/5611d54f08aec422d1171d3b/download

[17] Liu, Z., Sun, T. and Sang, G., n.d. An Algorithm of Association Rules Mining in Large Databases Based on Sampling. 2nd ed. [ebook] Dalian Maritime University: Dalian Maritime University, p.7. Available: http://article.nadiapub.com/IJDTA/vol6_no6/9.pdf

[18] "GitHub - satishrath185/Market-Basket-Analysis: Market Basket Analysis of Grocery and Retail dataset by Association Rules mining", GitHub, 2021. [Online]. Available: https://github.com/satishrath185/Market-Basket-Analysis.