Bilkent University, Department of Computer Engineering

CS425 – Algorithms for Web-Scale Data

Project Report

GROUP 11

MEMBERS

Umut AK, CS, ID: 21300931

Can AVCI, CS, ID: 21301193

Nursena KAL, CS, ID: 21501322

Ahmet Batu ORHAN, CS, ID: 21402365

24 DECEMBER 2018

Last Modified: 24 December 2018

Contents

Contents i

1. Introduction 1

2. Data 1

   a. Dataset Finding 1

   b. Dataset Generating 1

3. Algorithms 1

   a. A-Priori Algorithm 1

   b. PCY A-Priori Algorithm 2

4. Results 3

5. Conclusion 4

6. References 4

1. Introduction

This report was prepared by the members of the Group 11 of CS425. This report explains the project in detail. First, the dataset and how datasets were found and generated is given. Then, algorithms and results of those algorithms are detailly explained.

2. Data

In that subsection, how datasets were found and how the additional data was generated is detailly explained.

a. Dataset Finding

To start the project first is to be taken was finding datasets. Since people are not sharing what they bought in the last years, methods like crawling cannot be used to create a real dataset. Because of that, datasets must be found. To do that, as a team, websites like UCI Machine Learning, Philippe Fournier Viger, Kaggle, Data World is detailly searched, and datasets related to the project is downloaded.

b. Dataset Generating

As mentioned in the previous subsection, since web crawling or similar methods cannot be used in our project, already prepared datasets are used. However, those datasets must be changed to see the differences between what happens if a larger dataset is used or a dataset which has a higher number of items per basket or a dataset which has a different standard deviation. To test those cases, a small chunk of codes written in Java. Those codes can generate new datasets for the project which have a higher or lower standard derivation, higher or lower items per basket, higher or lower number of baskets, higher or lower number of items and so on. Plus, to generate some of the cases, like changing a number of items per basket or changing standard derivation, a real dataset that found in the web is used. Basically, the code generates an artificial data by checking the real data.

3. Algorithms

In that part, using algorithms which are A-Priori and the PCY algorithm, an extension of A-Priori Algorithm, is detailly explained.

a. A-Priori Algorithm

A-Priori is an algorithm to find frequent itemsets and their association rules. Its applications are mainly in recommendation systems. For example, when buying an item in Amazon, you may see a discount on an item which is related to the item you were about to buy.

The power of A-Priori comes from the fact that for many datasets we can eliminate almost all sets from candidacy and decrease the count of items we are working on. A-Priori is a two-pass algorithm. First, pass counts each item and finds the frequent ones. The high frequency is determined by a threshold called minimum support value. The second pass works only on frequent items that were found to be at least as frequent as minimum support value. The second pass counts the pairs of items since we eliminated less frequent items in pass one it only pairs the frequent ones. To find items of three or more we need more passes for each.

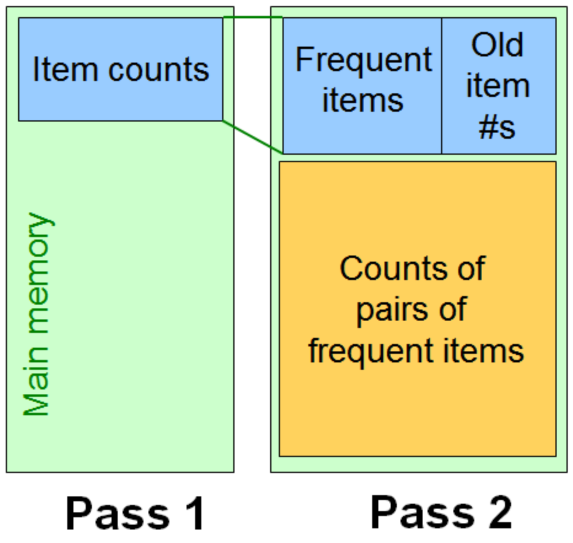
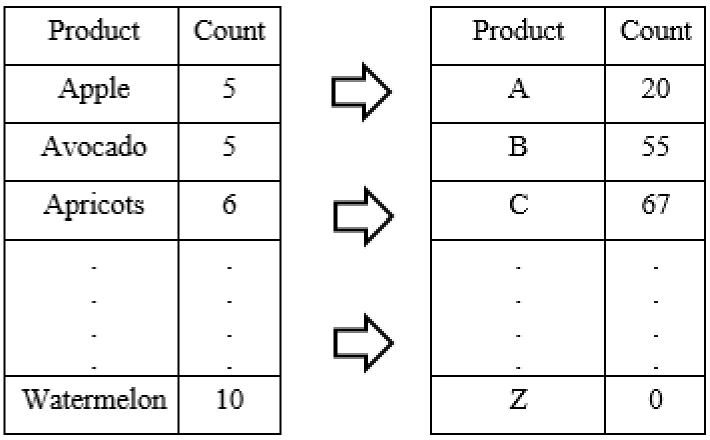


Figure 1: Memory usage of Apriori Ref MMD

As it can be seen we only use a small percentage of memory in pass1. Park-Chen-Yu algorithm extended the A-Priori to get used for the idle memory in pass1 and relief the work of pass two where memory is mainly used.

b. PCY A-Priori Algorithm

A Park-Chen-Yu (PCY) Algorithm, is a slightly modified version of the A-Priori algorithm. The algorithm uses hashing during the first pass. By that way, the algorithm reduces the number of candidate pairs. Those pairs are considered in the second pass. The key insight of the PCY algorithm is given below, with an example first.



Product	Count
Apple	5
Avocado	5
Apricots	6
. . . .	. . . .
Watermelon	10

Product	Count
A	20
B	55
C	67
. . . .	. . . .
Z	0

Table 1: Insight of the PCY Algorithm

Assume each item in the market must be counted one by one, to see if there is an item below the threshold limit and let's say threshold limit is 50. Rather than counting each item in the dataset, the number of items that started with each letter of the alphabet can be counted. If the products start with the letter 'A' is below the limit, then all the products must be below the threshold level.

As mentioned, this was the key insight. The idea is that, instead of applying it to single items, we apply it to pairs. Like instead of counting the {apple}, {banana} and {orange} we count the pairs like {apple, banana}, {apple, orange} and {banana, orange}.

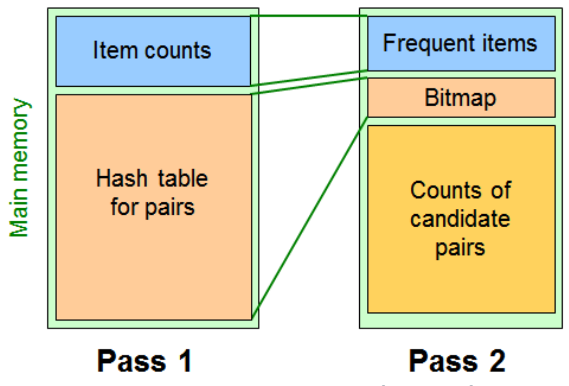


Figure 2: Main memory usage of PCY Ref MMD

On the second pass, we use a bitmap to map the hash table. Instead of using 4bytes for an integer count value, the memory used for keeping counts decreases by 1/32. A bit-vector is one if the pair is bigger than minimum support value and 0 if it is less.

Buckets are only needed to be counted until they reach the minimum support level. Number of buckets should ideally be quarter of main memory

4. Results

The result was generated according to the algorithm that was written by using the PCY logic. While testing the algorithm, bucket size, minimum support values, item per basket value, standard derivation and other things were changed. According to our results, there is a threshold value for bucket size. In other words, when the bucket size increased, the average time of the compilation of the algorithm decreases. However, after a threshold value for bucket size, the average time of the compilation of the algorithm started to increase. However, changing bucket size is also affecting the percentage of a number of false positives.

Moreover, when the minimum support value is increased, the average time of the compilation of the algorithm is increased. For instance, when we increased the minimum support value from 5000 to 10000, time increased from 8 to 9. Also, when we decrease the minimum support value, the number of frequent itemsets of size two is also increased.

Frequent Itemset of size 2, with minimum support, is equal to 10000:

- 1534,1943

Frequent Itemset of size 2, with minimum support, is equal to 5000:

- 1126,2183
- 1142,2363
- 1142,349
- 1215,225
- 1394,1989
- 1534,1582
- 1534,1943
- 1816,1834
- 1943,90
- 2363,349

Moreover, memory errors were gotten while doing the tests. Those memory errors affected the number of tests that accomplished successfully.

5. Conclusion

To conclude, as a team, we first find and then generated datasets according to the real ones. Later on, we took some online lectures on web to understand the logic of A-Priori and PCY algorithms. After that, as a team again, we started implemented the psudo code of the desired algorithms. Then, we tried to implement the desired algorithms by using Python and Java. Lastly, we try to understand the behaviour of the code by using different datasets that was generated by using the Java code and previous datasets.

6. References

https://archive.ics.uci.edu/ml/datasets/online+retail

http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php

https://www.kaggle.com/chiranjivdas09/ta-feng-grocery-dataset/home

https://toolbox.google.com/datasetsearch

http://www.mmds.org/

https://www.dunnhumby.com