Frequent Itemsets Mining for Recommender Systems

CMSC 5741 Group 7 - Project Report

Ziwen LU
1155155161@link.cuhk.edu.hk
Department of Information Engineering
The Chinese University of Hong Kong

Yan WU
1155148594@link.cuhk.edu.hk
Department of Information Engineering
The Chinese University of Hong Kong

Yaling ZHANG
1155147233@link.cuhk.edu.hk
Department of Information Engineering
The Chinese University of Hong Kong

Bowen FAN
1155155953@link.cuhk.edu.hk
Department of Information Engineering
The Chinese University of Hong Kong

ABSTRACT

Online shopping has become the major trend of consumer behavior and we have witnessed fierce growth of e-commerce platforms like Amazon, Taobao, Ebay, etc. Meanwhile, many competitors are entering the online retailing industry to capitalize the historical opportunity of such a major trend. It can be expected that the fight for e-commerce market share will only become more brutal than ever, and therefore having a competitive e-commerce platform becomes the most critical factor in this fight. Besides providing products with appropriate price and good quality, the most dominant e-commerce platforms have become more intelligent, in a way that they understand their customers more than their competitors, which led to the purpose of our project. This project focused on understanding customers, by analyzing the purchase history of customers to find new products that they might be interested in and recommending the new products to customers through a recommendation system. We collected three months of retailing data from a famous online retailer, which recorded more than 100 million customer behaviours (more than 20 gigabytes of data), including purchasing, adding to cart, viewing specific products and so on. With such mega data, we built lists of frequent itemsets of size 1 to 4, in order to understand what sets of products were often viewed or bought together. Furthermore, we developed an application which asked users to input their ID and then automatically recommend products to the users based on products they have viewed, purchased, etc. We have successfully developed and tested our recommendation system app, which could give the user recommendation from frequent itemsets of various sizes based on their preferences. Further details of the algorithm and implementation would be discussed in this paper.

KEYWORDS

Frequent Itemsets, Hadoop, MapReduce, Recommender Systems

1 INTRODUCTION

The retail industry took a distinguished turn with the flourish of online shopping. With the speed and convenience of online retail, it has become easier for consumers to get what they want when they want it. Moreover, due to the influence of COVID-19, people are more inclined to shop online recently. However, the online shopping industry can be cutthroat. This is why understanding online shopping statistics is more important now than ever to get ahead of the competition.

1.1 Motivation

Customers always have great expectations from brands they are interested in. In this case, providing customized product recommendation to different customers will increase the retailers' competitiveness. Therefore effective recommendation system which filters a large scale of information becomes necessary. However, the ongoing rapid expansion of online shopping and the diversity of customers' interests makes it difficult to conduct recommendation.

To further illustrate such difficulty, firstly, the number of digital buyers worldwide keeps climbing every year. In 2019, an estimated 1.92 billion people purchased goods or services online. During the same year, e-retail sales surpassed 3.5 trillion U.S. dollars worldwide, and according to the latest calculations, e-commerce growth will accelerate even further in the future [1]. Secondly, customers' shopping behavior can be affected by different factors such as regions, personal habits, global events, etc. For example, In a world-wide statistics, the top online categories for purchasing are fashion (61%), travel (59%), books and music (49%), IT (47%), and events (45%), but in the Asia Pacific, the most popular online industries are packed groceries (40%), home care (37%), fresh groceries (35%), and video gaming (30%) [2], see in Figure1.

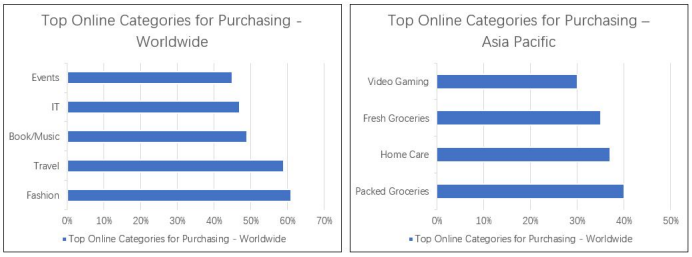

Figure 1: Statistics of Retail

Moreover, retail platforms have undergone an unprecedented global traffic increase between January 2019 and June 2020, surpassing even holiday season traffic peaks. Overall, retail websites generated

almost 22 billion visits in June 2020, up from 16.07 billion global visits in January 2020. This is of course due to the COVID-19 which has forced millions of people to stay at home in order to stop the spread of the virus [3].

Considering all these factors above, it does make sense to analyze a large scale of data set and extract inspiring advice for nowadays' recommendation systems.

1.2 Objective

Many other academic researches tend to focus on improving the efficiency or capability of recommender systems. Based on several topics of CMSC5741, this project aims to present more dimensions in regard to item recommendations. For instance, what brands should be recommended to shoppers, at what range of price shoppers are inclined to buy a product, when should recommendations be given to shoppers according to their shopping preference, what kind of similar items should be recommended, etc. The proposed recommender system can help users to find their purchasing habits according to their previous purchasing history as well as fulfil the functionality of doing recommendation.

The procedure of constructing the recommender system can be described as follows: first, we collected data from E-commerce platforms. Then, we pre-processed the data and stored them into a new CSV file as input. After that, we conducted three rounds MapReduce to calculate itemset frequency with implementation of SON algorithm. Finally, we built a user-friendly interface to recommend products according to users' feedback. Figure 2 shows the data processing process.

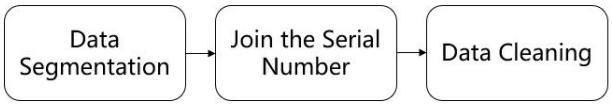

Figure 2: Procedure of Data Processing

1.3 Report Structure

This project report mainly introduces how to build a recommendation system by filtering frequent itemsets. Section 1 specifies the motivation and the objective of the project. Section 2 introduces current research direction in E-commerce recommender system. Section 3 describes how the recommender system is constructed based on frequent itemset mining. Section 4 shows the implementation result of the project. Section 5 summarizes and gives some discussions of the project.

2 RELATED WORK

[4] Li et al. gave a conventional idea of classifying subsets of products based on association rules. The next step it utilized collaborative filtering to do recommendation. The combination of these two ideas eliminated the influence of the other side so that a more accurate result could be provided.

[5] Nnadi suggested a method of relevant set correlation to do clustering in recommender system. The analysis of purchasing history through calculating the similarities of different items and different itemsets provided a more accurate and faster way to recommend the top-N products a consumer will buy with the highest probability.

[6] Moses et al. raised up a concept of scrutable recommender system. The methods of Fuzzy decision tree was included in the design. The system would update the recommendation generator to correct the system in accordance with users' unsatisfied manner.

[7] Friedman et al. proposed an interesting way to extract feature through the vectors which could substitute the recommended products semantically. Models for learning similar vectors could also be applied to translate the consumer browsing history to a simple sentence. Obviously,it dealt with the high dimension problem and proved to be highly efficient.

3 METHODOLOGY

3.1 System Design

Figure 3 shows the whole picture of our proposed recommender system. The system consists of four parts: data processing, decision trees for recommendation, database management and interactive interface. The recommendation part will give different advice for users, which enhances the probability to give a suitable recommendation.

3.2 Dataset

3.2.1 *Data Collection.*

This project uses E-commerce behavior data from multi-category store available on [8] as datasets. The datasets contain 285 million users' purchasing events from a large multi-category online store for a duration of 3 months, dated from February 2020 to April 2020. Each dataset is shown in the CSV format.The statistics of the datasets is summarized in Table 1.

Table 1: Statistics of Datasets

Dataset	Size	Records
2020-Feb	7.14GB	55,318,566
2020-Mar	7.28GB	56,341,242
2020-Apr	8.62GB	66,589,269
Total Size of Datasets	23.04GB	
Total No. of Records	178,249,077	

3.2.2 *Data Pre-processing.*

Since the project will not use the entire dataset to proceed, data in the CSV file are loaded into the R environment according to their tags and wait for selection. At the same time, the 'row' column is added to facilitate the subsequent index. For data cleaning, the strategy of deleting data if there exists missing values in different tags. The final processing result is in the format of [row, user_id, product_id] with a size of 4.78GB.

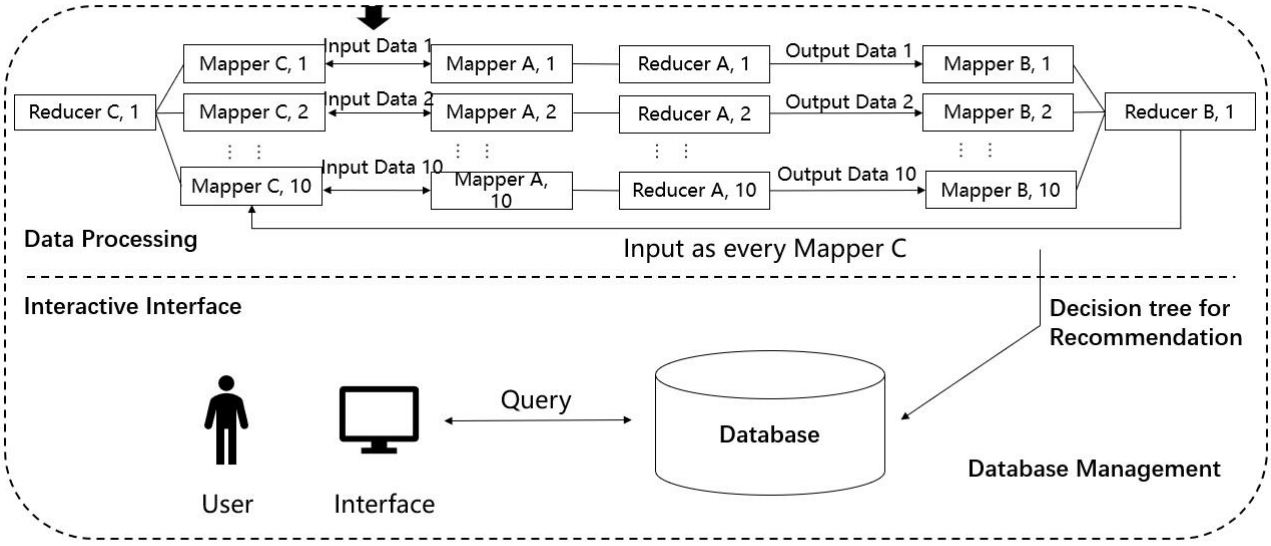

Figure 3: Recommender System Architecture

3.3 Frequent Itemset Mining

3.3.1 A-priori Algorithm.

The purpose of association rule mining is to find the hidden relationship between items. In order to reduce the generation time of frequent itemsets, some sets which are completely impossible to be frequent itemsets can be eliminated based on two laws of A-priori.

A-priori's Law 1: if a set is a frequent itemset, then all its subsets are frequent itemsets. Example: suppose a set A, b is a frequent itemset, that is, the number of times A and b appear in a record at the same time is greater than or equal to the minimum support, then its subsets A,b must appear more than or equal to minimum support, that is, its subsets are frequent itemsets.

A-priori's Law 2: if a set is not a frequent itemset, then all its supersets are not frequent itemsets. Example: suppose the set A is not a frequent itemset, that is, the number of times A appears is less than minimum support, then any superset such as A,b must appear less than minimum support, so its superset must not be a frequent itemset.

A-priori algorithm adopts the iterative method. Firstly, the candidate itemsets and the corresponding support are calculated, and the 1-term itemsets which have support lower than the support threshold are pruned to get the frequent 1-term itemsets. Then the algorithm combines the remaining frequent 1-term itemsets to get the candidate frequent 2-term itemsets. After that, it filters out the candidate frequent 2-term itemsets which are lower than the support threshold to get the real frequent 2-term itemsets. And so on, it iterates until the frequent K+1 itemsets cannot be found. The corresponding frequent K-term itemsets are the output of the algorithm.

3.3.2 SON Algorithm.

SON Algorithm is a partition algorithm based on A-priori. It scans the item datasets twice. In the first scan, all candiate frequent itemsets will be generated. In the second scan, each itemset obtained from the last scan will be counted to get its support, and the final frequent itemsets will be obtained according to the minimum support. The process of the algorithm is divided into two stages. In stage one, the algorithm divides the whole item datasets into non overlapping partitions logically, and each partition is independent of each other. The local frequent itemsets on the partition are calculated separately for each partition. Finally, the local frequent itemsets caculated on all partitions are summed up to get a global candidate itemsets. In the previous stage, the support of each candidate itemset in the whole item dataset is calculated, and then whether the minimum support meets the condition can be determined, so as to obtain the final result. The logic of SON algorithm is such that if an itemset is globally frequent, the itemset must be locally frequent at least in one partition.

3.3.3 Implementation.

We decided to use the SON algorithm with MapReduce implementation to acquire frequent itemsets from large amounts of data. A sequence of three MapReduce tasks were designed to accomplish the frequent itemset mining task.

The first MapReduce was designed to produce basket lists such that every user ID represented one basket, and products bought by the user became the corresponding items in the basket.

Mapper A, which is the first mapper, would read in the retail store data, and then sort the data according to the user ID. Finally, it outputted a tuple in the format of [user ID, product ID] for every retail record. With the output from the mapper, reducer A would collect all the product IDs and assign them to the corresponding user IDs. To achieve such goal, reducer A would create a dictionary with keys being all the user IDs. Whenever it received the output from mapper A, it would append the product IDs to the right key, see in Figure 4.

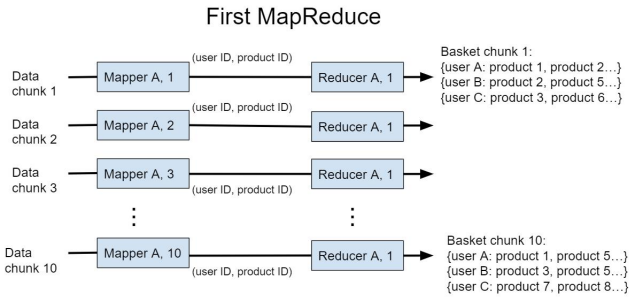

Figure 4: Illustration of the first MapReduce

The SON algorithm started with the second MapReduce. Given the size of the datasets, along with trial and error for parameter tuning, we decided that 600 would be a reasonable support threshold for finding frequent itemsets. Then we split the basket lists produced by the first MapReduce to 10 chunks, each chunk with a support threshold of 60. The second MapReduce would then apply the A-priori algorithm to each chunk to find all the frequent itemset candidates.

To be specific, 10 mapper B were assigned to handle the baskets of the 10 chunks. Each mapper read in the basket lists from one chunk, and calculated the support for every single product within that chunk and filtered out frequent itemsets of size 1. After that, it produced every possible combination of frequent itemsets size 1 and calculated support for all combinations to find frequent itemsets of size 2. The same process iterated until we found the frequent itemsets of size 4.

Now, all frequent itemset candidates of the 10 mappers would be passed to one reducer B, which checked repetitiveness of the incoming frequent itemset candidates and removed repeated frequent itemsets, therefore outputting a complete list of frequent itemset candidates, see in Figure 5.

The third MapReduce was responsible for finding the true frequent itemsets from the candidates and returning all true frequent itemsets with their support. To be detailed, the basket lists produced by the first MapReduce were split into ten chunks, each of which would be fed into one Mapper C. The complete list of frequent itemset candidates was also passed into all 10 Mapper C, which would then calculate the support for the list candidates within their

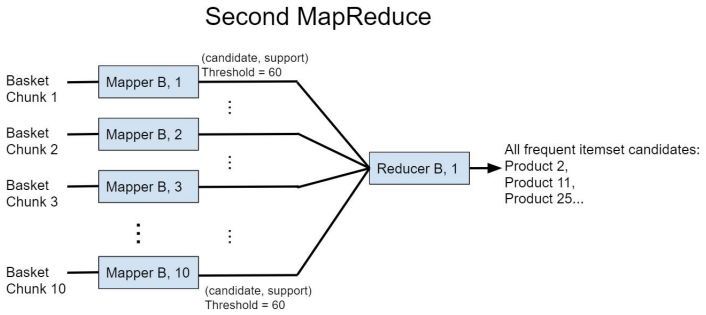

Figure 5: Illustration of the second MapReduce

corresponding chunks. Upon completion of calculation, all 10 mapper C would output tuples of the form [candidate frequent itemset, support] to one reducer C for support summation. Reducer C would sum all supports of the same candidate from the ten chunks, and filter out terms with support no less than 600 and return the true frequent itemsets with their corresponding support. The final output marked the end of the frequent itemset algorithm, see in Figure 6.

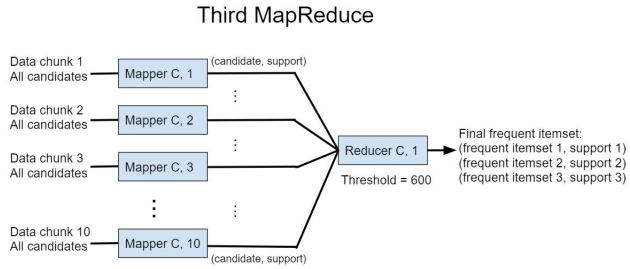

Figure 6: Illustration of the third MapReduce

3.4 Recommend Algorithm

3.4.1 Overview of main existing recommendation system algorithms.

Normally there are many mature algorithms for recommender systems, such as Collaborative filtering (CF) and its modifications, which is one of the most commonly used recommendation algorithms. Its core idea is to find the similarity of users or their preferences according to previously given information. Another algorithm is Matrix decomposition for recommendations, which is a very elegant recommendation algorithm because usually, when it comes to matrix decomposition, we don't give much thought to what items are going to stay in the columns and rows of the resulting matrices. Furthermore, some rather simple algorithms such as Clustering is more suitable for small systems. And due to a huge leap in growth on neural networks in the last 10 years, Deep learning approach for recommendations is also applied in a wide range of applications and are gradually replacing traditional ML methods.

3.4.2 Drawbacks of main existing recommendation system algorithms.

Considering our Recommender system generated by A-Priori algorithm, machined learning methods may be infeasible here. Two main reasons are sufficient to support this conclusion. First, once the data sets become very large, the memory requirements for the presentation of data grows exponentially. Second, due to the fact that the size of the available purchase database is usually growing linearly in the actual business scenes while the data grow exponentially with the size of the product assortment, they are incompatible with each other.

3.4.3 *A different approach.*

In this case, we were pursuing a very different approach which was based on the frequent itemsets and following steps were based on the frequent itemsets for generating a decision tree. In the following sections, we would display a sample to clarify this approach more explicitly.

3.4.4 Frequent Itemsets generation. Based on the A-Priori algorithm and Hadoop platform, we generated frequent itemsets with size from 1 to 4. Thanks to the distributed system, this process was fast. Final results was large, so let us check them step by step. Table 1 represented few parts of user-basket pairs which was extracted from original data. To make it simple to understand, raw data had been changed with explicit characters.

Table 2: Example of Purchasing History Datasets

User ID	Baskets
001	A,B,C,D
002	A,D
003	A,C
004	B,D
005	C

Based on our MapReduce implementation, we defined a fixed support threshold and made use of multiple rounds of MapReduce jobs to find frequent itemsets with size from 1 to 4. Table 3 below showed a sample of these data.

Table 3: Itemsets and their support threshold

Item set	UserID	Support threshold
{}	001, 002, 003, 004, 005	5
A	001, 002, 003	3
B	001, 004	2
C	001, 003, 005	3
D	001, 002, 004	3
A,B	001	1
A,C	001, 003	2
A,D	001, 002	2
B,C	001	1
B,D	001, 004	1
B,C	001	1
C,D	001	1
A,B.C	001	1
A,B,D	001	1
B,C,D	001	1
A,B,C,D	001	1

3.4.5 Frequent Itemsets analysis.

The actual support was much larger than what was shown in above samples, to simplify the explanation, we defined the itemset as a frequent itemset if and only if its support threshold exceeded or equal to 2, in this case, all single item and empty set would be considered as frequent item sets, but normally we exclude the empty set. With the utilization of frequent item sets of size from 1 to 4, it was now sufficient to implement the recommendation. One basic thought was to directly recommend these itemsets, however, considering that one frequent itemset in the whole purchasing history dataset might not be applicable to a specific user such as item A in the sample is not even in the basket of user 005. In this case, it was necessary to realize a precise recommendation.

3.4.6 Adjacency of frequent itemsets.

To achieve a precise practical personalized recommendation, a general idea was to organize these frequent itemsets together with original purchase history dataset. In this section, we mainly talked about the adjacency of frequent itemsets. Based on the A-Priori algorithm, it was distinct that frequent itemsets with size 2 must conclude part of single frequent items, and this pattern was scalable for item sets of greater size. Figure 6 shows an intuitive structure of this pattern. In this figure, we can see the connectivity between different sizes of frequent item sets.

$$\{\}(with\ support\ 5) \Longrightarrow \begin{cases} \{A\}(with\ support\ 3) & \Longrightarrow \begin{cases} \{A,C\}(with\ support\ 2) \\ \{A,D\}(with\ support\ 2) \end{cases} \\ \{B\}(with\ support\ 2) & \\ \{C\}(with\ support\ 3) & \Longrightarrow \{A,C\}(with\ support\ 2) \\ \{D\}(with\ support\ 3) & \Longrightarrow \{A,D\}(with\ support\ 2) \end{cases}$$

Figure 7: Illustration of Item Sets Adjacency

3.4.7 Decision trees deduced from frequent itemsets.

A Decision Tree is a predictive model expressed as a recursive partition of the feature space to subspaces that constitute a basis

for prediction. A Decision Tree is rooted directed tree and all other nodes are terminal nodes or leaves of the Decision Trees. In our case, nodes with outgoing edges are the previous frequent item sets. Decision Trees classify using a set of hierarchical decisions on the features. The decisions made at previous nodes are the split criterion, which is trying to find a frequent item sets with their sizes all one more than the previous item set's size. The root of this decision tree can be generated from another criterion, which is trying to find the 5 most frequent single items in the whole purchase history dataset and simultaneously in the specific user's purchase histories, this criterion aims to ensure the generalizability as well as the accuracy. In the basis of the two criteria, we do not need to make use of complex Machine Learning methodologies, it would be simple and efficient to make a precise personalized recommendation. Furthermore, decision trees can be conveniently induced, exchanged, and visualized by many tools, which makes it convenient to analyze the final results.

3.4.8 *Mechanism of recommendation.*

This recommender system would satisfy one user's personalized recommendation one at a time. In brief, one user entered in his user ID and then 5 most frequent items which had been viewed by him would show up. Based on his own choice of whether this recommendation needs to be continued, for example, if he choose one of the single item and wants to be recommended more items, all frequent item pairs including the previous chosen item would show up. This process would stop upon the item sets with size of 4 showed up. Algorithm 1 showed a pseudo code of this process, to analyze the time complexity of this algorithm, assuming that lines in frequent item sets table is M and lines in original data set is N, therefore the inner join procedure would take $O(M * N)$, other queries would take a linear time. In this case, this process's time complexity is much lower than square increasing time complexity algorithms such as Collaborative Filtering(CF), which is based on the number of user, while in our case, the number of frequent item sets with an one-to-one correspondence with users must be much lower than the number of users.

Algorithm 1: Personalized recommendation algorithm

```
Data: All frequent item sets from size 1 to 4 and original
      purchase history dataset
Result: Personalized items recommendation
Input: userid, previous recommended items, original
       dataset
Output: each round's recommendation item category code
while user decides to get more recommendations do
    if no recommendations before then
        read userid;
        if userid exists in the purchase history dataset then
            inner join single frequent items table and
             original dataset;
            output the five most frequent single items;
        else
            directly represent the five most single frequent
             items in whole dataset;
        end
    else
        read userid, previous recommended items;
        inner join frequent items table and original dataset;
        output the recommendation items' category codes in
         this procedure;
        this procedure's output becomes the next
         procedure's input;
    end
end
```

3.5 Interface

In this project, an interface was designed through PyQt5 to compile all the achievements.

3.5.1 *Layout of Interface.*

The interface can be mainly divided into two forms to imitate shopping websites in real life. The left form is designed to show user's personal information links like browsing history and purchasing history. The right form fulfills the function of login and recommendation. To meet the user-friendly demands, some buttons are prepared to let the user to decide whether he/she will continue to accept the recommendation.

3.5.2 *Mechanism of Interface.*

We put the processed frequent itemsets mining result into MySQL database and recall the needed information through query. If the user input his/her user_id, then he/she gets the privilege to get the recommendation. Our recommendation policy is: first, the user gets the frequent itemsets from the top-5 products in his/her previous purchasing history; then, if he/she is not satisfied with our recommendation, our system will provide another recommendation from its 2-term, 3-term, 4-term frequent itemsets to further enhance the user trust of the system.

4 RESULT

4.1 Experimental Environment

The project proposes a decision tree-based frequent itemsets mining method in recommender system. We implemented the algorithm in MapReduce structure using Hadoop distributed system. Then, to make it more user-friendly, we designed an interface to present our results. Listed below is the environment of our recommender system.

- R 3.6
- Hadoop2.9.2
- MySQL 8.0.22.0
- PyQt5

4.2 Interface

A simple user guide is provided to better illustrate our proposed recommender system.

- Step 1: User needs to input his/her user_id to login and click the item button in the left form to ask for recommendation.
- Step 2: Recommender system will first recommend what he/she has bought according to the browse history.

- Step 3: User will click the button in the bottom part of the right form to ask for more recommendation if he/she is not satisfied with the first round recommendation.
- Step 4: Recommender system will search from the frequent item pair to do next round recommendation until the quaternion pair is recommended.

Figure 8 shows the entire design of the system interface which is consisted of five labels to display our recommendation based on frequent itemsets mining. Four buttons are provided for users to get recommendation again if he/she is not satisfied with the first recommendation.

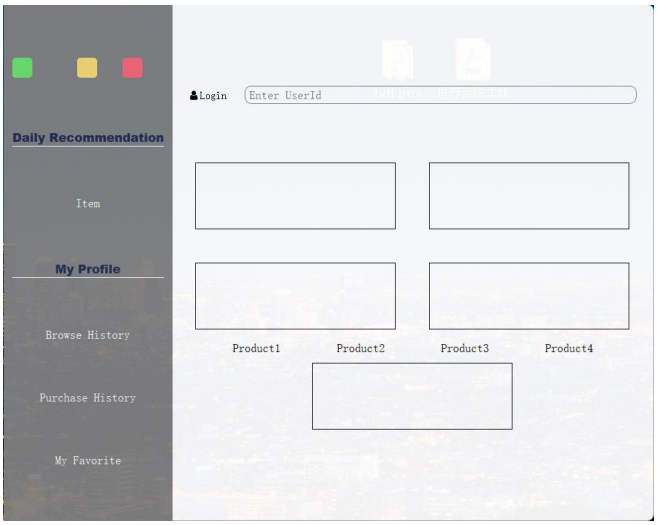

Figure 8: Layout of Interface

5 DISCUSSION

In this project, we proposed a frequent itemset mining based recommender system. The system uses decision trees to do recommendation by taking users' statisfaction into account. We compiled the frequent itemset mining results into a database then performed query to do personalized recommendation based on user_id.

It is worth mentioning that our recommender system still has some problems. If the user is a new comer, we can only do one-time recommendation based on mining previous users' browsing and purchasing history. In this way, the recommendation will only focus on the most frequent item bought by previous users which may not suit the new comer's taste. Also, if a user only browse or purchase one product in the website, he/she will not be available for multiple recommendation due to our recommendation policy. More work should be done in the future.

Since the dataset contains some information relevant to the purchasing time of the user, we can make use of it to analyze their consumer behaviours. Based on this kind of data, the recommender system can be adjusted more precisely among different times of one day. Meanwhile, more analysis on which kind of product or which brand is the most popular one among users can be conducted.

REFERENCES

[1] Nestor Gilbert. 74 compelling online shopping statistics: 2020 data analysis market share, 2020.

[2] J. Clement. E-commerce worldwide - statistics facts, 2020.

[3] J. Clement. Covid-19 impact on global retail e-commerce site traffic 2019-2020, 2020.

[4] Yu Li, Zuo Meiyun, and Bo Yang. Analysis and design of e-supermarket shopping recommender system. In *Proceedings of the 7th International Conference on Electronic Commerce*, ICEC '05, page 777–779, New York, NY, USA, 2005. Association for Computing Machinery.

[5] Nkechi J. Nnadi. Applying relevant set correlation clustering to multi-criteria recommender systems. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, page 401–404, New York, NY, USA, 2009. Association for Computing Machinery.

[6] Sharon J. Moses and L. D. Dhinesh Babu. A scrutable algorithm for enhancing the efficiency of recommender systems using fuzzy decision tree. In *Proceedings of the International Conference on Advances in Information Communication Technology amp; Computing*, AICTC '16, New York, NY, USA, 2016. Association for Computing Machinery.

[7] Asnat Greenstein-Messica, Lior Rokach, and Michael Friedman. Session-based recommendations using item embedding. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces*, IUI '17, page 629–633, New York, NY, USA, 2017. Association for Computing Machinery.

[8] Michael Kechinov. E-commerce behavior data from multi category store, 2019.