សាកលវិទ្យាល័យ ភូមិន្ទភ្នំពេញ

ROYAL UNIVERSITY OF PHNOM PENH

ការបង្កើតកម្មវិធីប្រព័ន្ធអនុសាសន៍
ដោយប្រើប្រាស់ក្បួនដោះស្រាយ Apriori

Recommendation System Application Development
by using Association Analysis Apriori Algorithm

A thesis
In Partial Fulfilment of the Requirement for the Degree of
Master Program of Information Technology Engineering

SAO Kimsong

ឆ្នាំ ២០១៩

សាកលវិទ្យាល័យ ភូមិន្ទភ្នំពេញ

ROYAL UNIVERSITY OF PHNOM PENH

ការបង្កើតកម្មវិធីប្រព័ន្ធអនុសាសន៍ ដោយប្រើប្រាស់ ក្បួនដោះស្រាយ Apriori

Recommendation System Application Development by using Association Analysis Apriori Algorithm

A thesis
In Partial Fulfilment of the Requirement for the Degree of
Master Program of Information Technology Engineering

SAO Kimsong

Examination committee: **Dr.**
Dr.
Mr.

ឆ្នាំ ២០១៩

Abstract in Khmer

សព្វថ្ងៃនេះភាគច្រើននៃក្រុមហ៊ុនអាជីវកម្មនៅលើពិភពលោកកំពុងតែប្រើបច្ចេកវិទ្យាសម្រាប់ដំណោះស្រាយបញ្ហា និង ការគ្រប់គ្រងរបស់ពួកគេ។ ឧទាហរណ៍ ធនាគារ និង ស្ថាប័នហិរញ្ញវត្ថុនានានឹងផ្តល់សិទ្ធិដល់អតិថិជនរបស់ពួកគេ ឱ្យធ្វើប្រតិបត្តិការហិរញ្ញវត្ថុដូចជាការផ្ទេរប្រាក់ និង ផ្ទេរប្រាក់តាមវើលវ៉ាយវ៉ែប (WWW)។ ហាងលក់រាយដូចជាក្រុមហ៊ុន វ៉ាលម៉ាត (Wall-Mart) ក៏ប្រើឧបករណ៍អេឡិចត្រូនិចដើម្បីស្តេនទំនិញដែលពួកគេទិញពីអតិថិជនហើយរាល់ព័ត៌មានប្រតិបត្តិការទាំងអស់ត្រូវបានរក្សាទុកនៅក្នុងប្រព័ន្ធគ្រប់គ្រងទិន្នន័យ។ យុទ្ធសាស្ត្រ Data Mining ដូចជា Association Rules និង Frequent Patterns ត្រូវបានប្រើជាទូទៅដើម្បីវាយតម្លៃឥរិយាបថទិញរបស់អ្នកប្រើប្រាស់។ ថ្មីៗនេះ Mining Rules ត្រូវបានកែប្រែដើម្បីប្រើប្រាស់ក្នុងប្រព័ន្ធអនុសាសន៍ (Recommendation System)។

គោលបំណងនៃការស្រាវជ្រាវនេះ គឺស្នើឱ្យមានស្ថាបត្យកម្មនៃប្រព័ន្ធការវិភាគទំនិញសម្រាប់ផ្តល់អនុសាសន៍ ការបង្កើត និងធ្វើការពិសោធន៍ប្រព័ន្ធអនុសាសន៍ដោយប្រើប្រាស់ Association Analysis។ នៅក្នុងគម្រោងរបស់យើង យើងមានគោលបំណងណែនាំផលិតផលដល់អតិថិជនឬអ្នកប្រើប្រាស់ដោយផ្អែកលើប្រវត្តិនៃការបញ្ជារទិញ។ Apriori Algorithm ជាក្បួនដោះស្រាយដ៏សំខាន់ដែលត្រូវបានប្រើនៅក្នុងការស្រាវជ្រាវរបស់យើង។ Apriori Algorithm គឺជាក្បួនដោះស្រាយទូទៅដែលអាចត្រូវបានប្រើដោយអ្នកអភិវឌ្ឍន៍ស្របតាមតម្រូវការរបស់ពួកគេហើយអនុវត្តវានៅក្នុងគម្រោងរបស់ពួគេ។

និក្ខេបបទនេះក៏បានបង្ហាញពីការពិនិត្យមើលឡើងវិញនូវប្រភេទនៃប្រព័ន្ធផ្តល់អនុសាសន៍និងវិធីសាស្ត្រផ្តល់អនុសាសន៍ផ្សេងៗគ្នាដែលត្រូវបានចាត់ថ្នាក់ជាចម្បងជាបីប្រភេទគឺ ការច្រោះសហការគ្នា ការច្រោះមាតិកា និង ការច្រោះកូនកាត់។ វិធីសាស្ត្រនីមួយៗមានចំណុចខ្លាំងនិងចំណុចខ្សោយរបស់វាដែលទាក់ទងនឹងដែន។

ចុងបញ្ចប់ យើងបានធ្វើការពិសោធន៍លើមូលដ្ឋានទិន្នន័យពិតរបស់ផ្សារទំនើបចំនួនពីរ។ យើងបានបង្ហាញលទ្ធផលនៃពេលវេលាឆ្លើយតបនៃ Apriori Algorithm និងដោះស្រាយលទ្ធផលពិសោធន៍។

ពាក្យគន្លឹះ៖ Recommendation System, Data Mining, Apriori Algorithm, Frequent Itemset, Association Rule.

Abstract in English

Today most companies and corporations around the world are using technology solutions for their work and business environment. For example, banks and financial institutions may allow their customers to make financial transactions, such as transfers and transfer of money via the World Wide Web. Retail stores like Wal-Mart also use electronic devices to scan items they buy from consumers and all transaction information is stored in the database. Data mining strategies such as association rules and frequent patterns are commonly used to evaluate the purchasing behavior of consumers. Recently, the mining rules of association have been modified to be used in recommendation systems.

The research aims at proposing architecture of association item analysis for the recommendation system and developed and conducted an experiment of recommendation system by using association analysis. In our project, we aim at recommending products to the customer or user based on the transaction purchase history. Apriori Algorithm is the main algorithm used in our research. Apriori Algorithm is the general algorithm which can be used by developers according to their need and implement it in their projects.

This thesis also presented a review of the categories of recommender systems and different recommendation methods that are mainly classified into three categories: collaborative filtering, content-based filtering, and hybrid filtering. Each method has its strengths and weaknesses that relate to the domain.

Finally, we have conducted an experiment on two supermarket real-life database. We also showed the result of the response time of the Apriori algorithm and discussed experiment results.

Key words: Recommendation System, Data Mining, Apriori Algorithm, Frequent Itemset, Association Rule.

SUPERVISOR'S RESEARCH SUPERVISION STATEMENT

TO WHOM IT MAY CONCERN

Name of program: Master of Information Technology Engineering
Name of candidate: SAO Kimsong

Title of research thesis: Recommendation System Application Development by using Association Analysis Apriori Algorithm.

This is to certify that the research carried out for the above titled master's research report was completed by the above-named candidate under my direct supervision. This thesis material has not been used for any other degree. I played the following part in the preparation of this research thesis conceptual design and methodological advices, idea organization, and thesis format advice.

Supervisor's name: Dr. SRUN Sovila

Supervisor's signature: …….

Date: …….

CANDIDATE'S STATEMENT

TO WHOM IT MAY CONCERN

This is to certify that the research report that I, SAO Kimsong, hereby present entitled "Recommendation System Application Development by using Association Analysis Apriori Algorithm" for the degree of Master of Science at the Royal University of Phnom Penh is entirely my own work and, furthermore, that it has not been used to fulfill the requirements of any other qualification in whole or in part, at this or any other University or equivalent institution.

Candidate's signature: ……

Date: ……

Signed by Supervisor: Dr. SRUN Sovila

Supervisor's signature: ……...

Date: ……

ACKNOWLEDGEMENTS

It is my great pleasure to express my utmost gratitude and heartfelt thanks to my deeply respected supervisor Dr. SRUN Sovila for his treasured guidance and motivation in making it possible complete this thesis. The valuable suggestions he gave throughout the whole process and willingness to guide me without any hesitation is honestly acknowledged.

I would like to thank my colleagues at Blue Technology Co., Ltd who aided me directly or indirectly throughout my research work.

SAO Kimsong

December 2019

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 Background of the Study

Today most companies and corporations around the world are using technology solutions for their work and business environment. Their business transactions are done using computer systems. For example, banks and financial institutions may allow their customers to make financial transactions, such as transfers and transfer of money via the World Wide Web. Retail stores such as Wal-Mart also use electronic devices to scan consumer items they buy, and all transaction information is stored in the database. Amazon lets customers buy and sell their books and other products through its website, and customers can provide reviews through ratings and comments. The customers' feedback is stored in the database as well. In the form of reviews or comments, Amazon allows customers to buy and sell books and other articles through its website (Greg Linden,Brent Smith,and Jeremy York, 2003). The whole customer input is also stored in the database. Through data centers, a huge amount of this data is stored. Another example is Netflix, which enables customers to rate the films they watch and store the feedback information. These companies and companies contain huge amounts of data on their databases and data warehouses. It is important to analyze this enormous amount of data to gain useful information.

The review of input data such as reviews in places like Amazon and Netflix provides these businesses and consumers meaningful information at the same time. For example, in order to recommend such movies, Netflix analyzes film ratings of customers (Farhin Mansur, Vibha Patel, Mihir Patel, 2017). Amazon can also research the profile of a customer and evaluate the reviews given by the customer to recommend books and other things to him or her. All these types of recommendations are made through what is called recommendation systems.

Recommended system (RS), one of the most powerful and useful tools in the digital world today. Recommender Systems (RSs) are software tools and techniques that provide recommendations for user-friendly products. The recommendations given aim to help their users in various decision-making processes, such as what things to purchase, what music to listen to, or what news to read. RSs have proven to be a valuable way for online users to cope with the overload of information and has become one of the most powerful and

popular electronic commerce tools (Francesco Ricci, Lior Rokach and Bracha Shapira, 2011). Think of the fact that Amazon recommends you books they think you might like; Amazon might use a Recommender Program behind the curtains to make effective use. Therefore, various techniques have been proposed for the generation of recommendations and many of them have also been successfully deployed in commercial environments over the past decade.

Consequently, various techniques for the generation of recommendations have been developed and many of them have also been successfully deployed in commercial environments over the past decade. The approach to content deals with item profiles and user profiles and is designed to recommend text-based items (Francesco Ricci, Lior Rokach and Bracha Shapira, 2011). For commercial areas, the collaborative filtering technique is commonly used. Amazon uses the shared search technique to recommend to its customers books and other items (G. Adomavicius, A. Tuzhilin, 2005). RSs based on collaborative filtering recommend items to a user based on similar items rated by some other users and share the same preferences of items or products with the target consumer and other users (G. Adomavicius, A. Tuzhilin, 2005). In order to recommend products, demographic approach suggest systems use demographic information such as gender, age and date of birth of the respective users (Marko Balabanovic,Yoav Shoham, 1997). The hybrid approach was developed to address the shortcomings and disadvantages of the other approaches to recommendations (G. Adomavicius, A. Tuzhilin, 2005). The hybrid approach incorporates two or more approaches to guidance to remove the drawbacks of single approaches. Some studies show that hybrid approaches can be more effective than other approaches. (G. Adomavicius, A. Tuzhilin, 2005).

Data mining strategies such as association rules and frequent patterns are commonly used to evaluate the purchasing behavior of consumers (Jiawei Han, Micheline Kamber, Jian Pei, 2012). For instance, distributors can analyze the market basket in order to find out what customers have to do with marketing strategies by finding association rules and frequent item setups (Jiawei Han, Micheline Kamber, Jian Pei, 2012). Recently, the mining rules of association have been modified to be used in recommendation systems. For example, Bendakir and Ameur have proposed a course recommendation system based on association rules (N.,Bendakir, and E.,Aımeur, 2006). Also, Xizheng has proposed a personalized recommendation system using association rule mining and classification in e-commerce (Z., 2007).

1.2 Problem Statement

The analysis of shopping baskets has been very attractive to retailers in recent years. Advanced technology allowed them to collect information about and purchase from their customers. The implementation of electronic point-in-sales increased the use and use by market basket analysis of transactional data. The analysis of such information is extremely useful for understanding the purchasing behavior of retail businesses. Mining purchasing patterns allows retailers to better customize promotions and store settings. For every successful business, identifying purchasing rules is crucial. For mining of useful information on joint purchases and adjustment of promotion and advertising accordingly the transactional data is used. The well-known set of beer and diapers is just an example of an association rule found by data scientists.

1.3 Aim and Objectives of the Study

Our research aims to propose the association analysis architecture of association item analysis for the recommendation system. In this study, we develop and conduct experiments of recommendation system by using association analysis Apriori algorithm. Transactional data mining association guidelines will provide valuable information about co-occurrences and goods co-purchases.

1.4 Rationale of the Study

The rationale of the study lies the most successful application of data mining is the recommendation application. This study will be used to analyze transaction databases and look for patterns among existing customer transactions. These patterns are used to help make business decisions, such as, what to put on sale, how to design coupons, how to place merchandise on shelves in order to maximize the profit, and selecting the items required and associated together in a timely manner.

1.5 Limitation and Scope

The thesis work is mainly focused on the proposed architecture of association item analysis for the recommendation system and conducted an experiment of the performance of the Apriori algorithm using the real-world database.

1.6 Thesis Structure

Chapter 2 is the background and related work, and it provides necessary concepts, methods, and algorithms of association rules mining and recommendation systems. Chapter

3 presents our approach in detail, and it illustrates our proposed algorithm. Chapter 4 shows the experiments results of our proposed algorithm. Finally, the conclusion and the future work are presented in chapter 5.

CHAPTER 2
LITERATURE REVIEW

The aim of this chapter is to provide a thorough review of the research topics, areas and works discussed here. First, we carry out a brief but thorough in-depth study of association rule mining techniques and methods, followed by an overview of interestingness and performance, and issues of redundancy relevant to association rule mining. Finally, we look at the recommender process and the related cold-start problem. The analysis sets the stage for our work and the suggestion that has been made here.

2.1 Mining Frequent Patterns and Association Rules

The patterns frequently used in the data set are defined as patterns (Jiawei Han, Micheline Kamber, Jian Pei, 2012). Several items, for example bread, butter and milk, often common in a transaction are referred to as a frequent itemset. Mining in a frequent itemset enables us to detect the associations and correlations between items in large data sets (Jiawei Han, Micheline Kamber, Jian Pei, 2012). In many retail stores for example, large quantities of data are collected and stored for their databases. Such volumes of data can be collected to define interesting associations between these database documents, which can help business managers to make decisions such as cross-marketing, behavioral consumer buying research and catalog design (Jiawei Han, Micheline Kamber, Jian Pei, 2012).

2.1.1 Association Rules

Let's make a set of items $I = \{I_1, I_2, I_3, \dots I_m\}$. Let D be a collection of transaction in a database which T is a set of items such that $T \subseteq I$. The *TID* identifier is connected to each transaction in the database and allows A to be a subset of items. A transaction T contains A if and only if $A \subseteq T$ (Jiawei Han, Micheline Kamber, Jian Pei, 2012). An association rule is an implication of the form $A \Rightarrow B$, where $A \subset I, B \subset I$, and $A \cap B = \emptyset$ (Jiawei Han, Micheline Kamber, Jian Pei, 2012). The set of items A and B are called antecedent and consequent of the rule respectively. The rule $A \Rightarrow B$, holds in the set of database transactions D with support s, where s is the percentage of transactions in D that contain $A \cup B$ which means the probability $P(A \cup B)$ indicates that a transaction contains the union of set A and set B (Jiawei Han, Micheline Kamber, Jian Pei, 2012). In addition, the confidence c of the rule $A \Rightarrow B$ in the transaction set D is the percentage of transaction in D that containing A that also containing B too which means the conditional probability $P(A \mid B)$ (Jiawei Han, Micheline Kamber, Jian Pei, 2012). Therefore, the rules that satisfy

both a minimum support threshold and a minimum confidence threshold are called strong association rules (Jiawei Han, Micheline Kamber, Jian Pei, 2012).

Support and Confidence for Itemset A and B are represented by the following equations:

$$Support(A) = \frac{Transaction\ contianing\ A}{Total\ Transactions}$$

$$Confidence(A \Rightarrow B) = \frac{support(A \cup B)}{support(A)}$$

The key method of association rule mining is typically to find all common items and produce strong association rules (Jiawei Han, Micheline Kamber, Jian Pei, 2012). Association rule mining consists of 2 steps. The first is about to find all the frequent itemsets. And then generate association rules from the frequent itemsets.

2.1.2 Apriori Algorithm

The Apriori algorithm is a well-known algorithm that is used for mining frequent itemsets for association rules (Jiawei Han, Micheline Kamber, Jian Pei, 2012). It is an algorithm for efficient association rule discovery (Hegland, 2007). The algorithm was proposed by R. Agrawal and R. Srikant in 1994. The approach that is used in the Apriori algorithm is known as a level-wise search, where k-itemsets are used to explore (k+1)-itemsets (Rakesh Agrawal, Ramakrishnan Srikant*, 1994). This algorithm uses two steps "join" and "prune" to reduce the search space. It is an iterative approach to discover the most frequent itemsets.

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.

In the first iteration of the algorithm, each item is taken as a $1-itemsets$ candidate. The algorithm will count the occurrences of each item. Let there be some minimum support (e.g. 2). The set of $1-itemsets$ whose occurrence is satisfying the minimum support are determined. Only those candidates which count more than or equal to minimum support, are taken ahead for the next iteration and the others are pruned (Jiawei Han, Micheline Kamber, Jian Pei, 2012).

Next, $2-itemset$ frequent items with minimum support are discovered. For this in the join step, the $2-itemset$ is generated by forming a group of 2 by combining items with itself. The $2-itemset$ candidates are pruned using minimum support threshold value. Now the table will have $2-itemsets$ with minimum support only (Jiawei Han, Micheline Kamber, Jian Pei, 2012). The next iteration will form $3-itemsets$ using join and prune step. This iteration will follow antimonotone property where the subsets of 3-itemsets, that is the $2-itemset$ subsets of each group fall in min_sup. If all $2-itemset$ subsets are frequent then the superset will be frequent otherwise it is pruned. Next step will follow making $4-itemset$ by joining 3-itemset with itself and pruning if its subset does not meet the minimum support criteria. The algorithm is stopped when the most frequent itemset is achieved (Jiawei Han, Micheline Kamber, Jian Pei, 2012).

To illustrate the Apriori algorithm, let us assume that we have these five transactions in the database:

Table 1. Customer purchase histories.

TID	Items
T1	I1, I2, I3
T2	I2, I3, I4
T3	I4, I5
T4	I1, I2, I4
T5	I1, I2, I3, I5
T6	I1, I2, I3, I4

TID is the transaction ID, and *Items* are the items that are bought by the customers.

We use the Apriori algorithm to find frequent itemsets and generate the association rules that satisfy the minimum support *s* which is 50% and minimum confidence *c* which is 60%.

First, we generate all the candidates of one itemset C_1 as shown in the Table 2:

Table 2. The candidates of one itemset C_1.

Item	Support Count
I1	4
I2	5
I3	4
I4	4
I5	2

Next, we remove the items that do not satisfy the support count. Table 3 shows the frequent one itemset L_1:

Table 3. The frequent one itemset L1.

Item	Support Count
I1	4
I2	5
I3	4
I4	4

Next, we get all the candidates of two itemsets C_2 by applying the joint operation on L_1 ($C_2 = L_1 \boxtimes L_1$). Then, we remove the itemsets that do not satisfy the support count as shown in Table 4:

Table 4. The frequent two itemset L2.

Item	Support Count
I1, I2	4
I1, I3	3
I2, I3	4
I2, I4	3

Then, we do the joint operation again on $L_2$to get C_3. Next, we look for the subsets that are frequent. We remove itemset that it contains a subset that is not a frequent itemset. The frequent three itemsets $L_3$is showing in the Table 5:

Table 5. The frequent three itemsets L3.

Item	Support Count
I1, I2, I3	3

Since the L_3 contains only one set, we cannot do the joint operation on L_3. Thus, $C_4 = \emptyset$, so we stop. Then, we can list the association rules for example in the form of $buy(X, item1)\ buy(X, item2) \Rightarrow buy(X, item3)$ with its support s and confidence c.

Finally, we list the strong association rules that satisfy the minimum support s which is 60% and the minimum confidence c which is 80%. Table 6 shows the strong association rules:

Table 6. Strong Association Rules.

Rules	Confidence
$\{\boldsymbol{I1}, \boldsymbol{I2}\} \Rightarrow \{\boldsymbol{I3}\}$	$\frac{Support\ \{I1, I2, I3\}}{Support\ \{I1, I2\}} = \frac{3}{4} * 100\ =\ 75\%$
$\{\boldsymbol{I1}, \boldsymbol{I3}\} \Rightarrow \{\boldsymbol{I2}\}$	$\frac{Support\ \{I1, I2, I3\}}{Support\ \{I1, I3\}} = \frac{3}{3} * 100\ =\ 100\%$

$\{\boldsymbol{I2}, \boldsymbol{I3}\} \Rightarrow \{\boldsymbol{I1}\}$	$\frac{Support\ \{I1, I2, I3\}}{Support\ \{I2, I3\}} = \frac{3}{4} * 100\ =\ 100\%$
$\{\boldsymbol{I1}\} \Rightarrow \{\boldsymbol{I2}, \boldsymbol{I3}\}$	$\frac{Support\ \{I1, I2, I3\}}{Support\ \{I1\}} = \frac{3}{4} * 100\ =\ 75\%$
$\{\boldsymbol{I2}\} \Rightarrow \{\boldsymbol{I1}, \boldsymbol{I3}\}$	$\frac{Support\ \{I1, I2, I3\}}{Support\ \{I2\}} = \frac{3}{5} * 100\ =\ 60\%$
$\{\boldsymbol{I3}\} \Rightarrow \{\boldsymbol{I1}, \boldsymbol{I2}\}$	$\frac{Support\ \{I1, I2, I3\}}{Support\ \{I3\}} = \frac{3}{4} * 100\ =\ 75\%$

Apriori Algorithm – Pros

- Easy to understand and implement.
- Can use on large itemsets.

Apriori Algorithm – Cons

- At times, you need many candidate rules. It can become computationally expensive.
- It is also an expensive method to calculate support because the calculation must go through the entire database.

2.2 Recommendation System

The basic ideas and methods of recommendations systems are discussed in this section. Content based, collaborative filtering, demographic and hybrid methods are included. The section also explains the drawbacks of existing recommendation approaches.

2.2.1 Basic Concepts

To provide more formal definition of recommendation systems, let U be a set of all possible users, and let I be a set of all possible items (G. Adomavicius, A. Tuzhilin, 2005). In many e-commerce applications, the space U and I can be very large. Let f be a utility function that measures the usefulness of an item i to a user u such as $U \times I \rightarrow R$ where R is an ordered set of non-negative integers or real numbers (G. Adomavicius, A. Tuzhilin, 2005). Then, for each user $u \in U$, we want to choose an item $i_u \in I$ to maximize the user's utility as shown below (G. Adomavicius, A. Tuzhilin, 2005):

$$\forall\, u \in U, i_u = \arg\max f(u, i)$$

In the context of recommendation systems, the utility of an item is usually represented by a rating. For example, James gave the movie Spider Man a rating of 4 (out

of 5). Basically, the utility of an item indicates how a user liked an item (G. Adomavicius, A. Tuzhilin, 2005). Table 7 shows an example of (User × Item) rating matrix:

Table 7. shows an example of (User × Item) rating matrix.

User / Item	*Spider Man*	*Die Hard I*	*Die Hard II*	*The Flight*	*Bad Boys II*
James	5	7	6	2	∅
Jessica	2	∅	5	∅	9
John	7	∅	3	5	1
Zack	4	6	10	∅	∅
Sara	∅	7	8	3	∅

The table above shows the ratings for each movie that the users have watched (out of 10). ∅ indicates the movie has not been rated yet by the user. Therefore, the goal of recommendation systems is to predict unrated items. Based on that predicated ratings, the recommendation systems will be able to select some items with highest predicted ratings and recommend them to the user (G. Adomavicius, A. Tuzhilin, 2005).

2.2.2 Content-Based Recommendation

The user rating items and the recommender system should understand the common features of the products that have been rated by the consumer in the past in content-based recommendations systems. The program then suggests products that are close to the preferences and tastes of the consumer (G. Adomavicius, A. Tuzhilin, 2005). For example, a content-based approach in a film recommendation system attempts to understand the common characteristics of the films that have been highly rated by the consumer, such as stars, guidelines, genres, etc. The program then recommends movies that match the tastes of the consumer (G. Adomavicius, A. Tuzhilin, 2005).

Content-based method constraints:

- Over-Specialization: An approach based on content tends to suggest items close to the products previously classified by the same user (G. Adomavicius, A. Tuzhilin, 2005). For example, for an article in sports or technology, a person who is interested in business articles receives little recommendation (Marko Balabanovic,Yoav Shoham, 1997).

- New User Problem: If a new user comes into the program, it does not have a user profile and no rating products yet are available. Therefore, the program cannot make accurate recommendations (G. Adomavicius, A. Tuzhilin, 2005).

2.2.2 Collaborative Filtering Recommendation

For e-commerce, collaborative filtering approaches are commonly used. (Greg Linden,Brent Smith,and Jeremy York, 2003). In many implementations, such as Amazon and Netflix, they have been successful (J. L.,Herlocker, J. A.,Konstan, A.,Borchers, and J.,Riedl, 1999). It is a popular technique for reducing the overload of information. In a collaborative filtering approach, Amazon recommends books to its clients. A shared filtering recommendation framework recommends items for a user based on the similar items selected by other users (G. Adomavicius, A. Tuzhilin, 2005). The program seeks objects with common interests to ask users for other applications. For example, the system identifies a group of users in movie recommendation systems that are focused on the collective filtering method who have similar preferences to a query user. Instead, the program recommends the films that those users have rated highly to the target consumer in the past (G. Adomavicius, A. Tuzhilin, 2005).

Collaborative filtering approaches are grouped into two general categories:

- Memory-based approaches: They use the complete collection of the items classified to make recommendations or predictions.
- Model-based approaches: They help systems to learn how to recognize patterns in data sets to make recommendations or forecasts (X.,Su, and T. M.,Khoshgoftaar, 2009).

Limitation of Collaborative Filtering Approaches:

- New User Problem: Collaborative filtering has the same problem as the method based on content that is new users accessing the system (G. Adomavicius, A. Tuzhilin, 2005). To make recommendations to a user, the program must be aware of the user's expectations from the ratings the user makes (G. Adomavicius, A. Tuzhilin, 2005). Because the consumer is new to the program, she / he has not yet rated products (G. Adomavicius, A. Tuzhilin, 2005).
- New Item Problem: The systems will include approved items so that users can suggest those items. On joining the systems, a new item has not yet been

reviewed by users. So, the systems won't be able to recommend it to the users.

- Sparsity: Sparsity is an important issue for collective filtering approaches. Inside the recommendation system, the total number of ratings is significant.
- Scalability: In many realistic collaborative recommendation filtering systems, the number of users and objects within the program is increasing rapidly.

2.2.3 Demographic Based Approach

A demographic recommendation system recommends products for the user based on the demographic information of the user such as gender, age and date of birth (B.,Amini, R.,Ibrahim, and M.S.,Othman, 2011). Depending on their demographic characteristics, the demographic approach divides users into groups. The program would, for example, position the users who belong to a certain zip code into one category. Only, the 18- to 25-year-old users will be in one category. The demographic based recommendation systems presume that users in the same party or category have the same values and preferences (B.,Amini, R.,Ibrahim, and M.S.,Othman, 2011). The demographic method monitors the users ' purchase or score actions within the same party or category. If a new user joins the program, the system will first position the user into a category based on demographic information provided by the user. The program will then recommend products or things to the user based on the other users in the group's purchasing or ranking behavior (Francesco Ricci, Lior Rokach and Bracha Shapira, 2011).

Grundy was an early example of a recommendation system centered upon demographic information. The system is designed to recommend books to library guests based on their personal information collected from them through an interactive dialogue (R., 2002). A further recent example of a demographic-based recommendation system is LIFESTYLE FINDER (B., 1997). The system uses consumer analysis demographic groups to recommend a variety of products and services and collects user data through a short survey (R., 2002).

The demographic-based approach has its limits. The first weakness faced by the demographic system is how to classify the party or category to which the user belongs when the user is new to the system (B.,Amini, R.,Ibrahim, and M.S.,Othman, 2011). The second weakness is how the users' interests and preferences within the same community are defined

(B.,Amini, R.,Ibrahim, and M.S.,Othman, 2011). The third drawback of the quantitative method is that when demographic data are available to the system the demographic system works well. But, collecting this kind of data is not easy (S. Anand, B., 2006).

Therefore, due to the limitations of the demographic approach few recommendation systems use the demographic approach (S. Anand, B., 2006). In addition, the accuracy of demographic-based recommendation systems is less than those content-based recommendation systems or collaborative filtering systems (S. Anand, B., 2006).

2.2.4 Hybrid Approach

Content-based and collaborative approaches to filtering have been widely used in the commercial and research fields. However, in the previous bits, they have many limitations. The hybrid approach has therefore been implemented to overcome the drawbacks of the content-based and collaborative approaches to filtering (G. Adomavicius, A. Tuzhilin, 2005). Some recommendation systems combine two or more methods to perform better and remove some of the disadvantages of the approaches to pure recommendation systems.

2.3. Recommendation Systems based on Association Rule Mining

There are some recommendation systems that use association rules mining techniques have been introduced in the literature. They are applied to various application areas in the real world such as e-Learning systems, e-Commerce systems, and course recommendation systems.

Chellatamilan and Suresh presented an idea for building a recommendation system for the e-Learning system using Association Rules Mining to provide students with the best selection of learning materials and e-Learning resources (T.,Chellatamilan, R.,SURESH, 2011). Their idea is to gather data from students using a survey questionnaire in area of educational background, IT experience, technology accessibility, frequency of their study patterns, demographics data, etc. In addition, the system analyzes students' logs of a Learning Management System (LMS) Moodle. Then, they apply data mining tools such as association rules to find frequent itemsets. Association rule mining, distance metrics such as Jaccard measure, and cosine of the angle are used to construct the recommendation system (T.,Chellatamilan, R.,SURESH, 2011). This system is required to gather personal and background data from the users in the form of a survey questionnaire. This is a major step in this system, and it can be considered as a disadvantage of the proposed

recommendation system. Recommendation systems that require gathering data such as demographic data work well only if the data is available (B.,Amini, R.,Ibrahim, and M.S.,Othman, 2011). Thus, failure to provide such data can cause poor recommendations.

Our proposed framework does not require gathering information from users, such as demographic information, in order to provide recommendations which is an advantage over the system proposed by Chellatamilan and Sures.

In (Abhishek Saxena, Navneet K Gaur, 2015) researchers used a technique focused on the frequency of the collection of items. They used a "bottom-up" approach in which regular subsets were expanded one item at a time and evaluated against the data by groups of candidates. When no more effective extensions are found, the algorithm stops. Particularly important are pairs or larger sets of items that appear much more frequently than the items purchased individually would be expected.

The approach used in (Karandeep Singh Talwar, Abhishek Oraganti, Ninad Mahajan, Pravin Narsale, 2015) researchers used to exploit the transaction history. As Apriori is designed to operate on transaction databases and generate association rules, using a "bottom-up" approach where frequent subsets are extended one item at a time and groups of candidates (the candidate set includes all the frequent k-length item sets) are evaluated against the information. When no further effective extensions are found, the algorithm stops. They also introduced four major features: User Interface Element, Data Extraction, Web Application Mining and Pattern Recognition.

Seven steps to improving the Apriori algorithm are given in (Ranjan S G, Sandesh A Hegde, Sujay N, Swaroop S Rao, Padmini M S, 2017) writing. The first is to search the collection of opinion information and decide the item's support(s). After this generate L1 (Frequently one item set) and use Lk-1, join Lk-1 to generate the k-item set. The third they scan the candidate k item set and generate the support of each candidate k – item set. The fourth, add to frequent item set, until C=Null Set. The fifth, for each item in the frequent item set generate all non-empty subsets. The last one, for each non-empty subset determine the confidence. If confidence is greater than or equal to this specified confidence, then add to Strong Association Rule.

In (Abaya, 2012) author modified the Apriori algorithm by taking the set size, which is the number of items per transaction and the set size rate, which is the number of transactions with at least "set size" items. In support of the revised version, the average

results for both the execution time and the moving list yield 38 percent and 33 percent respectively. Nevertheless, the result is consistent with the original algorithm in some test data. It has been found that as the number of items per transaction decreases, the desirable outcome will be from the original algorithm since pruning candidate keys is closer to the first k+1 while implementing the modified one takes a lot of execution time since pruning begins with k(n) – 1 where n is the total set size with set size frequency > = minimum support.

In (Shadi AlZu'bi, Bilal Hawashin, Mohammad ElBes, Mahmoud Al-Ayyoub, 2018) author proposed a new, powerful recommender framework for user requirements based on the Apriori algorithm. They used a list of application qualifications data and the rules obtained. In (JinHyun Jooa, SangWon Bangb, GeunDuk Parka, 2016) a recommendation system was designed and implemented to evaluate consumer preferences and personal propensities by using association rule analysis and cooperative filtering to collect customer data on customer visits to NFC (Near Field Communication) firms. Using the data analysis results and distance information from GPS (Global Positioning System), the recommendation algorithm used in the proposed system recommended local businesses that people are highly likely to visit. Jiao Yabing (Yabing, 2013) proposes an improved algorithm of association rules, the classical Apriori algorithm. It verifies the improved algorithm, the results show that the improved algorithm is rational and efficient, it can obtain more data about value. Proposed in (Sagar Bhise, Prof. Sweta Kale, 2017) based on the Frequent Pattern growth algorithm. They concentrate on the algorithm of PFP production, divided into two phases, namely the phase of pre-processing and the phase of mining. In (S.O. Abdulsalam, K.S. Adewole, A.G. Akintola, M.A. Hambali, 2014) Apriori algorithm was presented for extracting valuable knowledge embedded in the database of a supermarket for market basket analysis. Data representing six (6) distinct products across thirty (30) unique transactions were generated from a well-structured transactional database representing the sales pattern of a supermarket store.

In (Greg Linden,Brent Smith,and Jeremy York, 2003) Amazon uses cooperative sorting item-to-item matches each of the purchased and valued items of the customer with similar items, then combines those similar items into a recommendation list. To evaluate the most similar match for an item, the algorithm generates a table of similar items by finding items that customers prefer to buy together. Amazon could build a product-to-product matrix by iterating through all item pairs and computing a similarity metric for

each pair. However, many product pairs have no common customers, and thus the approach is inefficient in terms of processing time and memory usage. The key to item-to-item collaborative filtering's scalability and performance is that it creates the expensive similar-items table offline. The algorithm's online component — looking up similar items for the user's purchases and ratings — scales independently of the catalog size or the total number of customers; it is dependent only on how many titles the user has purchased or rated. Thus, the algorithm is fast even for extremely large data sets.

In (C S Fatoni, E Utami, F W Wibowo, 2018) authors proposed a product recommendation system based on Apriori method for online store. The system design method used is Reuse-Based has 6 stages in the system design process, among others, the collection of system specification requirements, component requirement analysis, system specification modification, combining the system design with Reuse-Based, development of merger system, and system validation process. The concept approach allows for the retrieval of reusable components and depends on the size of components that can be reused and integrated with the concept of components in the software. It can be concluded that by applying the apriori algorithm, the system provides product recommendations to Online Store customers based on the trust value of a combination of products purchased at a given time period. Application of Apriori Method in this research is to find the most combination of items based on transaction data and then form the association pattern of item combination.

Bendakir and Aimeur proposed a course recommendation system based on association rules mining (N.,Bendakir, and E.,Aımeur, 2006). The system incorporates a data mining process with user ratings in recommendations. Specifically, the architecture of the system is divided into two phases: an off-line phase which consists of a data mining process, and an on-line phase for the interaction of the systems with its users. The off-line phase is used to extract association rules from the data, and the on-line phase uses the rules to infer course recommendations. The advantage of this system is to allow the user (student) to evaluate the previous recommendations, so the system can be enhanced, and the rules are updated as more evaluations of the previous recommendations are provided by the students. But this system has disadvantages; it does not make use of a student's academic background (N.,Bendakir, and E.,Aımeur, 2006). Additionally, this system was developed to fit a certain context of recommendation systems, which is a course recommendation system.

Aijaz Ahmad Sheikh, Tasleem Arif, and Majid Bashir Malik proposed Technique for Recommender System (Aijaz Ahmad Sheikh,Tasleem Arif, Majid Bashir Malik, 2018). In their proposed system have five steps such as Data Collection, Opinion Mining, Rating Fusion, and Recommender Process (Aijaz Ahmad Sheikh,Tasleem Arif, Majid Bashir Malik, 2018). In step Data Collection, they collection of user's opinions from E-Commerce websites shall be performed using any data extraction technique. - Opinion mining is a combination of text mining and natural language processing. It uses supervised and unsupervised methods to evaluates the opinions and classify them as negative or positive. The computed rating from reviews of the item shall be fused with the numerical or star ratings. In the recommender process they proposed to use KNN or any other similar approach for recommendation of items (Aijaz Ahmad Sheikh,Tasleem Arif, Majid Bashir Malik, 2018).

CHAPTER 3
METHODOLOGY

In this chapter, we provide the details and description of our proposed framework. We illustrate the use of the algorithm and how it works on the context of a recommendation system. Also, we give comparisons of the proposed framework with other recommendation methods.

3.1 System Overview

As we described in the chapter on literature review, recommendation systems are commonly used in applications for e-commerce. Recommendation systems are aimed at recommending products to a customer. The literature introduces various approaches to recommendation systems such as content-based approaches, collaborative filtering, demographic approaches, and hybrid approaches.

We propose a recommendation framework that integrates association rule mining with a frequent itemset generation. We use the Apriori algorithm to generate a set of association rules. The Apriori algorithm mines over the frequent sets to discover association rules.

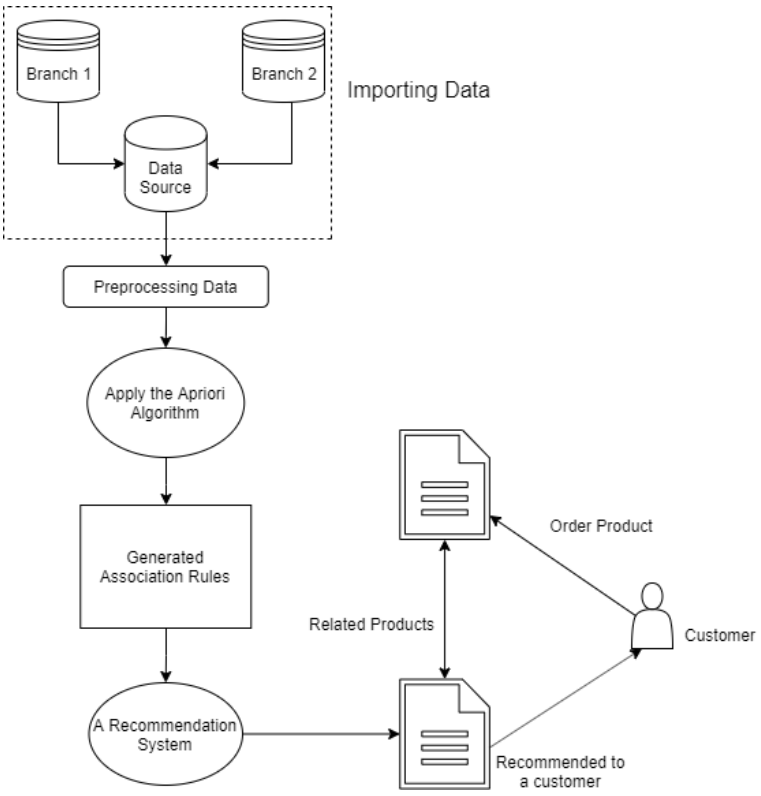


Figure 1. The diagram of the proposed framework.

The most important parameters in the Apriori algorithm are minimum support count and minimum confidence (R. Perego,P. Palmerini,S. Orlando). Generated association rules play an important role in our proposed recommendation framework, as illustrated in Figure 3.1.

Our proposed framework consists of main four parts. The first part is to download customer purchase transaction into our formation relational database (Data Source) from shop data. The second part is to do clean up data called preprocessing data. The third part is to apply the Apriori algorithm for generate frequent itemset. The third part is to apply the generated the association rule to recommend items for a customer.

3.2 Importing Data

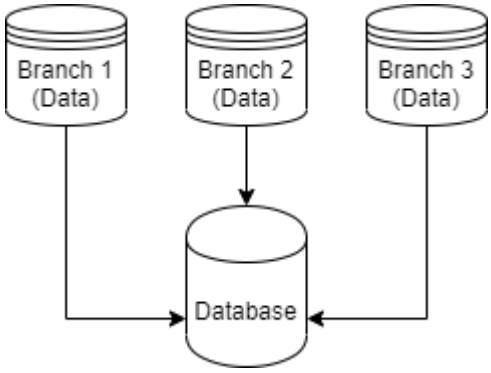


Figure 2. Importing Data Diagram

The initial process of system design is to collect customer purchase history data taken from any data source (Branch 1, Branch 2, …) and import into relational database. We used the MariaDB as a middleware for the relational database to store the transaction from which are import from any data source. In this step, we need to match the data source column with our relational database formation column called target columns. The source columns that provide must be the same size of target columns [Appendix A]. For example, for the transaction after imported shows in Table 8.

Table 8. Customer Purchase History.

TID	Items
T1	ESPRESSO
T1	SUGAR
T1	NEWSPAPER
T2	ESPRESSO
T2	SUGAR
T2	COLA

T3	ESPRESSO
T3	SUGAR
T4	CAPPUCCINO
T4	CIGARETTES
T5	CAPPUCCINO
T5	SUGAR
T6	CAPPUCCINO
T6	SUGAR
T6	SWEETS
T7	DECAF
T7	SUGAR
T7	CHEWING_GUMS
T8	DECAF
T8	SODA
T8	VINEGAR
T9	DECAF
T9	SUGAR
T9	CIGARETTES

3.3 Preprocessing Data

After we have been downloaded from shop data into relational database, we must clean up the data. In this step, we labeled the item as a number, for example ESPRESSO labeled 1, SUGAR labeled 2, NEWSPAPER labeled 3 etc. After matching the item label, we convert the historical transaction data into our algorithm formation. For example, for the historical transaction after clean-up shows in Table 9.

Table 9. Pattern of customer purchase history after cleaning.

TID	Items	Items Label
T1	ESPRESSO, SUGAR, NEWSPAPER	1, 2, 3
T2	ESPRESSO, SUGAR, COLA	1, 2, 4
T3	ESPRESSO, SUGAR	1, 2
T4	CAPPUCCINO, CIGARETTES	5, 6
T5	CAPPUCCINO, SUGAR	5, 2
T6	CAPPUCCINO, SUGAR, SWEETS	5, 2, 7

T7	DECAF, SUGAR, CHEWING_GUMS	8, 2, 9
T8	DECAF, SODA, VINEGAR	8, 10, 11
T9	DECAF, SUGAR, CIGARETTES	8, 2, 6

3.4 Frequent Itemset for the Apriori Algorithm

The customer purchase transaction pattern will take the number of transactions from each product item per transaction and the amount of transaction data then used to determine the itemset combination. The combination of 1-itemset is processing based on the data provided in table 3.1, the process of forming K_1 or called a combination of $1 - itemset$ with the minimum amount of support = 40%, by the formula in equation (1) (C S Fatoni, E Utami, F W Wibowo, 2018).

$$Support(A) = \frac{Number\ of\ transactions\ contraing\ A}{Total\ transaction} \qquad (1)$$

The squatter process in equation (1) obtains the data shown in table 10, for the support value of each product item.

Table 10. The support value of each product item.

Item	Label	Support
ESPRESSO	1	3/9 = 30%
SUGAR	2	7/9 = 77%
NEWSPAPER	3	1/9 = 11%
COLA	4	1/9 = 11%
CAPPUCCINO	5	3/9 = 30%
CIGARETTES	6	2/9 = 22%
SWEETS	7	1/9 = 11%
DECAF	8	3/9 = 30%
CHEWING_GUMS	9	1/9 = 11%
SODA	10	1/9 = 11%
VINEGAR	11	1/9 = 11%

The establishment of itemsets in table 10 with a minimum of 20% support can be found that meets the minimum standards of support on ESPRESSO product items, SUGAR, CAPPUCCINO, CIGARETTES, DECAF. Then from the result of combination formation 1 item will be done combination $2 - itemset$ as in table 11.

The Combination of 2 Items is processing based on data provided in table 10 items taken above the support value of each product item, the process of forming K_2 or called a combination of $2 - itemsets$ with minimum amount of support = 20%, by the formula in equation (2) (C S Fatoni, E Utami, F W Wibowo, 2018).

$$Support(A \cap B) = \frac{\sum transaction\ containing\ A\ \&\ B}{\sum transactions} \qquad (2)$$

Table 11. The support value of 2-itemsets.

Item	Label	Support
ESPRESSO, SUGAR	1, 2	3/9 = 30%
ESPRESSO, CAPPUCCINO	1, 5	0/9 = 0%
ESPRESSO, CIGARETTES	1, 6	0/9 = 0%
ESPRESSO, DECAF	1, 8	0/9 = 0%
SUGAR, CAPPUCCINO	2, 5	2/9 = 22%
SUGAR, CIGARETTES	2, 6	0/9 = 0%
SUGAR, DECAF	2, 8	2/9 = 22%
CAPPUCCINO, CIGARETTES	5, 6	1/9 = 11%
CAPPUCCINO, DECAF	5, 8	0/9 = 0%
DECAF, CIGARETTES	8, 6	0/9 = 0%

Combination 2 itemset with minimum 20% support can be seen combination of 2 itemset that meet minimum standard of support that is ESPRESSO, SUGAR with support of 30% and SUGAR, CAPPUCCINO with 22% support and SUGAR, DECAF with 22% support. From the result of the combination of 2 itemset will be done formation 3 itemset as in table 11. Combination of 3 Items is processed based on data provided in table 12 items taken above the support value of each product item, the formation process K_3 or called with a combination of 3 itemsets with minimum amount of support = 20%, by the formula in equation (3) (Francesco Ricci, Lior Rokach and Bracha Shapira, 2011).

$$Support(A, B, C) = \frac{\sum transaction\ containing\ A,B,and\ C}{\sum transactions} \qquad (3)$$

Table 12. The support value of 3-itemset.

Item	Label	Support
ESPRESSO, SUGAR, CAPPUCCINO	1, 2, 5	0/9 = 0%
ESPRESSO, SUGAR, DECAF	1, 2, 8	0/9 = 0%

SUGAR, CAPPUCCINO, DECAF	2, 5, 8	0/9 = 0%
ESPRESSO, CAPPUCCINO, DECAF	1, 5, 8	0/9 = 0%

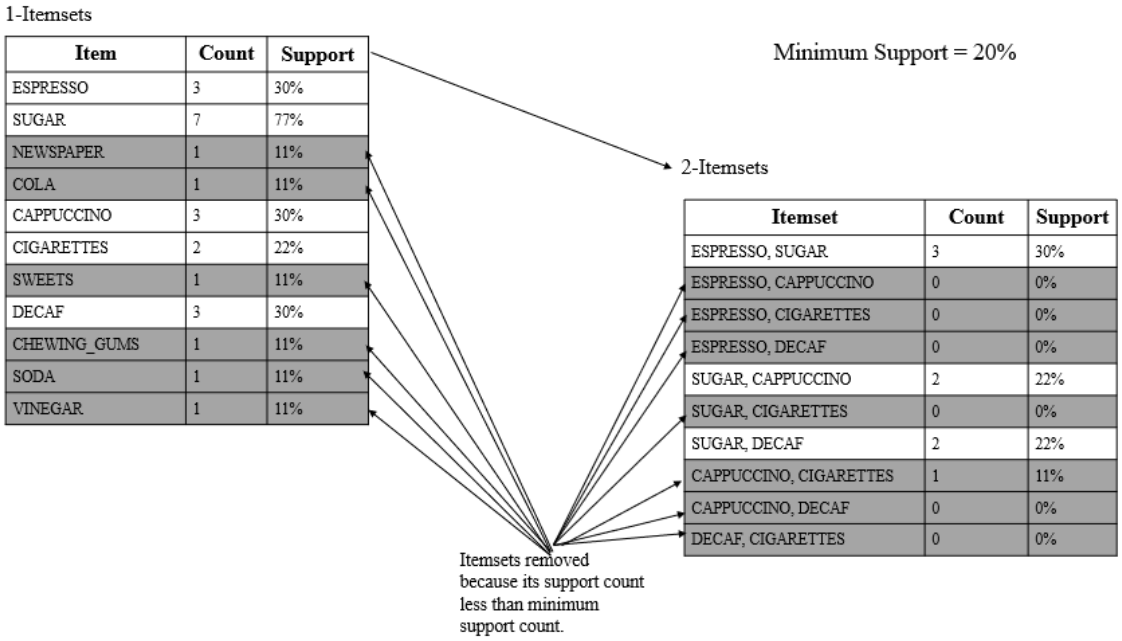


Figure 3. Illustration of high level of frequent itemset generation for the Apriori.

There is no frequent itemset can be seen combination of $3 - itemset$ that meet minimum standard of support.

The pseudocode for frequent itemset generation part of the Apriori algorithm is shown in Algorithm 3.1 [Appendix B]. Let C_k denote the set of candidates k-itemsets and F_k denote the set of frequent k-itemsets:

- The algorithm initially makes a single pass over the data set to determine the support of each item. Upon completion of this step, the set of all frequent 1 itemsets, F, will be known (steps 1 ,2 and 3).
- Next, the algorithm will iteratively generate new candidate k-itemsets (step 6). Candidate generation is implemented using a function called *created_ck* [Appendix C].
- To count the support of the candidates, the algorithm needs to make an additional pass over the data set (steps 7). The subset function is used to determine all the candidate itemsets in C_k that are contained in each transaction t. After counting their supports, the algorithm eliminates all candidate itemsets

whose support counts are less than minimum support. This step is implementation using a function called *scan_dataset* [Appendix D].

- The algorithm terminates when there are no new frequent itemsets generated, i.e., $F_k = \emptyset$ (step 9).

Algorithm 3.1: Frequent itemset generation of the Apriori Algorithm

---

1: $k = 1$
2: $C_1 = create\_c1(dataset)\{generate\ candidate\ for\ 1-itemset\}$
3: $F_1 = scan\_dataset(dataset, C_1, \min\_support)\ \{1-itemset \geq \min\ \_support\}$
4: $k = 2$
5: $repeat$
6: $\quad C_k = create\_ck\left(dataset\right)\{generate\ candidate\ for\ k-itemset\}$
7: $\quad F_k = scan\_dataset(dataset, C_k, \min\_support)\ \{k-itemset \geq \min\ \_support\}$
8: $\quad k = k + 1$
9: $until\ F_k = \emptyset$
10: $result = \bigcup F_k$

3.5 Association Rule Generation

After all the high frequency patterns are found, then the association rules that meet the minimum requirements for confidence by calculating the trust of the associative rule $A \Longrightarrow B$. Minimum Confidence = 60%. The confidence value of rule $A \Longrightarrow B$ is obtained.

$$Confidence(A \Longrightarrow B) = \frac{\sum transaction\ contain\ A\ \&\ B}{\sum transactions\ contain\ A} \qquad (4)$$

Table 13. Association Rule

Rule	Support	Confidence
$\{\boldsymbol{ESPRESSO}\} \Rightarrow \{\boldsymbol{SUGAR}\}$	3/9 = 33%	3/3 = 100%
$\{\boldsymbol{DECAF}\} \Rightarrow \{\boldsymbol{SUGAR}\}$	2/9 = 22%	2/3 = 66%
$\{\boldsymbol{CAPPUCCINO}\} \Rightarrow \{\boldsymbol{SUGAR}\}$	2/9 = 22%	2/3 = 66%

Based on Table 3.5, the products most often purchased by customers are espresso, decaf, cappuccino and sugar with knowledge of the products most often purchased by customers, then the company can develop strategies in determining the purchase of products to maintain product availability required by customers and also can adjust the location of the product based on the combination of product items formed.

The pseudocode for generation association rule part is shown in Algorithm 3.2 [Appendix E]. Note the similarity between the *rules_from_consequent* procedure given in Algorithm 3.3 and the frequent itemset generation procedure given in Algorithm 3.1. The

only difference is that, in rule generation, we do not have to make additional passes over the data set to compute the confidence of the candidate rules. Instead, we determine the confidence of each rule by using the support counts computed during frequent itemset generation.

Algorithm 3.2: Association rule generation of the Apriori Algorithm

---

1: $for\ each\ frequent\ k-itemset\ F_k\ do$
2: $H_1 = \{i \mid i \in F_k\}\ \{1-item\ consequents\ of\ the\ rule\}$
3: $rules = rules\_from\_consequent(F_k, H_1, \min\_confidence)$
4: $end\ for$

Algorithm 3.3: Procedure rule generation rules_from_consequent ($F_k, H_m, \min\ \_confidence$)

1: $k = |F_k|\ \{size\ of\ frequent\ itemset\}$
2: $m = H_m\ size\ of\ rule\ consequent$
3: $if\ k > m+1\ then$
4: $H_{m+1} = create\_ck(H_m)$
5: $for\ each\ h_{m+1} \in H_{m+1}$
6: $conf = \sigma(F_k) / \sigma(F_k - h_{m+1})$
7: $if\ conf \geq \min\ \_confidence\ then$
8: *Out the rule*
9: *end if*
10: $end\ for$
11: *end if*

3.6 Recommendation

After we have applied the Apriori algorithm and generation the association rule we got a list of strong rules. So, we do the recommendation for the customer or user in e-commerce application that we want. The figure 4 show you the expected result of recommendation system using Apriori algorithm.

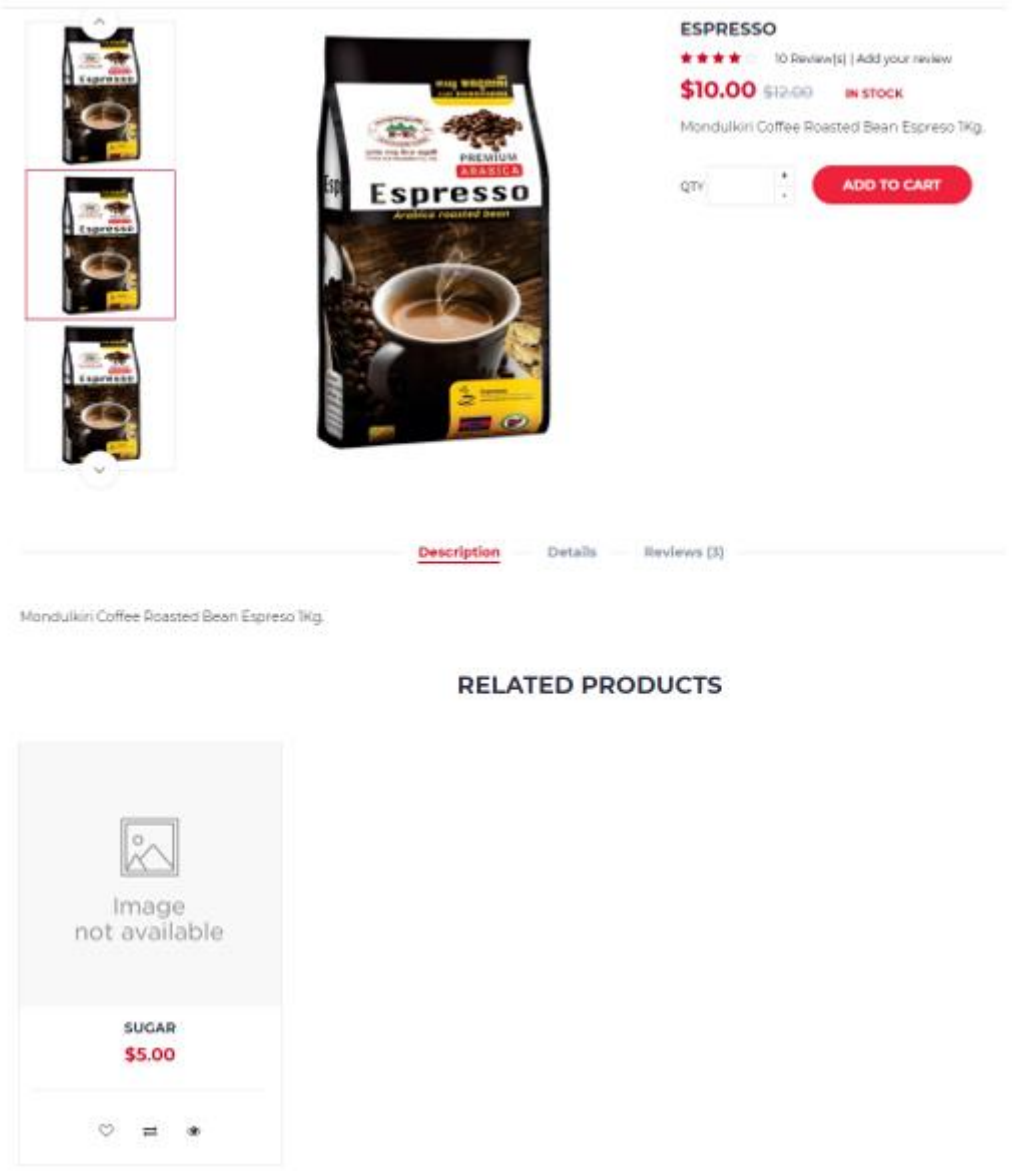


Figure 4. Recommended item to customer.

CHAPTER 4
EXPERIMENT

This chapter presents an experimental study of our proposed framework. The first section describes the experimental setup. The second section presents the experiment results. The last section summarizes our observation on the experiment results.

4.1 Environmental Setup

4.1.1 Datasets

We use two different of the data of Super Market. One it consists of 4,444 customer historical transaction and another one it consists of 189,919 transactions. The dataset is already cleaned up. There is no need to preprocess the datasets. But we have reformatted the dataset files to fit into our implementation of the proposed algorithm.

Table 14. Details of Datasets

Name	Total transactions	Average no of items per transactions
Dataset1	4,444	10
Dataset2	16,466	10

4.1.2 Hardware and Software

The following information is about the hardware that are used to conduct the experiments.

- Processor: Intel(R) Core (TM) i5-5200U CPU @ 2.20GHz, 2201Mhz, 2 Core(s), 4 Logical Processor(s).
- Available Ram: 16.00 GB
- System Model: Inspiron
- OS: Windows 10 version 1803

In order to generate the association rules, we have implemented the windows application for using Python 3.7 integrated with user interface PyQT5. PyQt5 is a comprehensive set of Python bindings for Qt v5. It is implemented as more than 35 extension modules and enables Python to be used as an alternative application development language to C++ on all supported platforms including iOS and Android (Riverbank, 2020). PyQt5 may also be embedded in C++ based applications to allow users of those applications to configure or enhance the functionality of those applications (Riverbank, 2020).

Additionally, we used Visual Studio Code to implement our proposed algorithm, and to write several associated functions.

4.1.3 Measures

In order to evaluate the performance and accuracy of Apriori algorithm after adding to our proposed system, we must evaluate it using some measures. The measures are (time and size) i.e., the period to retrieve the from shop data, and the size of data to be retrieved from the database and the response time of the frequent itemset generation for the Apriori Algorithm.

4.2 Experiments and Results

We used datasets that contains 4,444 and 16,466 transactions. The average number of items contained in a transaction is 4.6 and 3, while the variance is ± 5 items. The graph in Figure 5 and Figure 6 illustrates the performance of our implementation in means of response time (seconds) while the minimum support threshold varies from 0.1% up to 0.5%. We observe that while the minimum support decreases, the response time of the algorithm increases. This is expected, since lower values of minimum support result more frequent itemsets to be discovered and consequently more possible extensions.

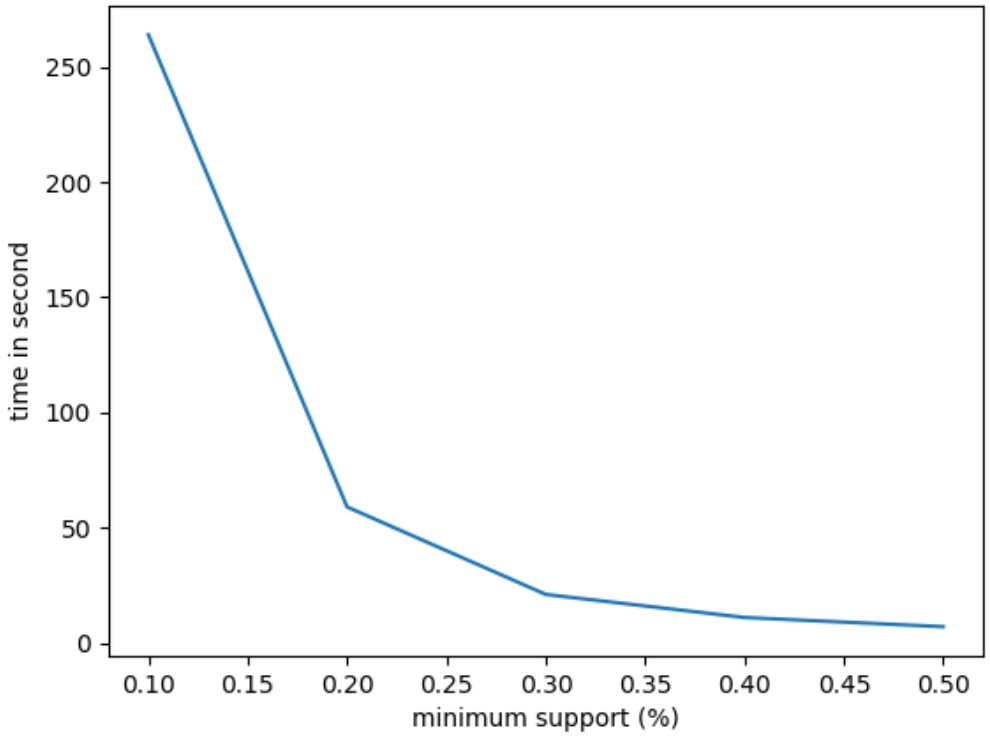


Figure 5. Response time of frequent itemset generation for Dataset1.

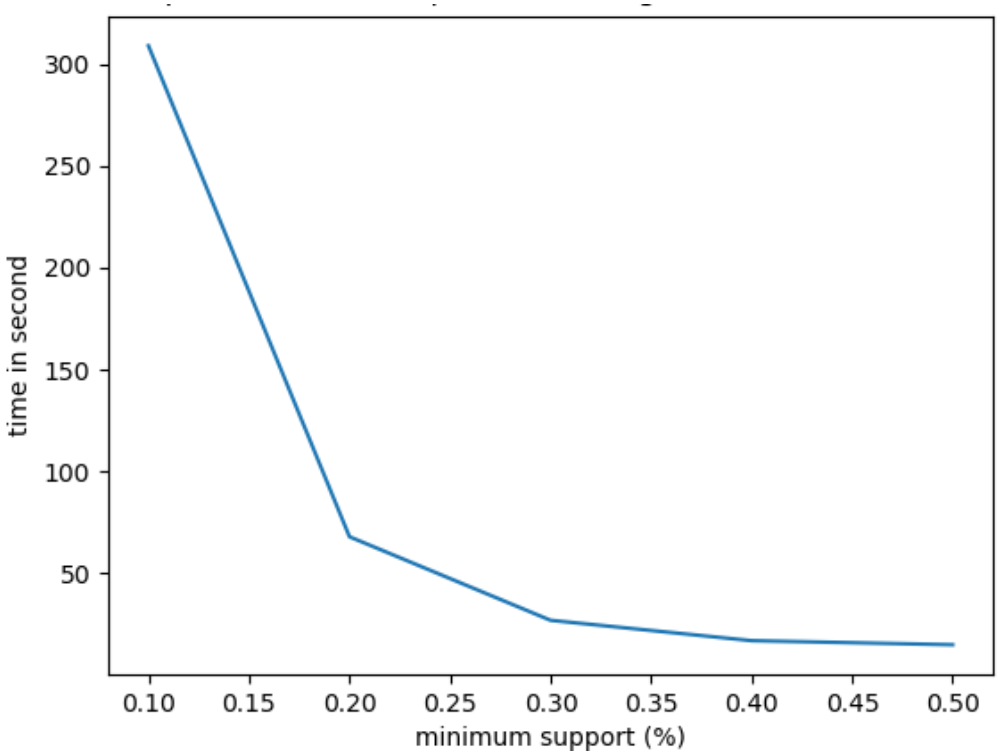


Figure 6. Response time of frequent itemset generation for Dataset2.

CHAPTER 5
CONCLUSIONS AND FUTURE WORKS

5.1 Conclusions

In this research, we proposed an architecture for association item analysis for recommendation system and we developed and conduct experiment of Recommendation System by using Association Analysis Apriori Algorithm. Our proposed architecture with constructing a recommender system which can understand the purchase behavior of the customers, by utilizing the historical transaction data, in retail store or e-commerce application.

We have done experiments on the proposed architecture the results that are extracted from those experiments show that our proposed framework can provide recommended a new item to customers by understanding historical transaction data.

5.2 Future Works

In the proposed framework, we must download data from shop's data into our relational database and run the Apriori algorithm on customer buying history. After we run the Apriori algorithm we got the list of association items.

Our future work is to create a library for to any e-commerce or retail store application for recommend the new item to customers by using association items from our proposed frameworks.

REFERENCES

Abaya, S. A. (2012). Association Rule Mining based on Apriori Algorithm in Minimizing Candidate Generation. *International Journal of Scientific & Engineering Research, 3*(7).

Abhishek Saxena, Navneet K Gaur. (2015). Frequent Item Set Based Recommendation using Apriori. *International Journal of Science, Engineering and Technology Research (IJSETR), 4*(5).

Aijaz Ahmad Sheikh,Tasleem Arif, Majid Bashir Malik. (2018). Framework for Opinion Based Product Recommender System. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*.

B., K. (1997). *Lifestyle finder: Intelligent user profiling using large-scale demographic data.* AI magazine.

B.,Amini, R.,Ibrahim, and M.S.,Othman. (2011). Discovering the impact of knowledge in recommender systems: A comparative study. *arXiv.*

C S Fatoni, E Utami, F W Wibowo. (2018). Online Store Product Recommendation System Uses Apriori Method. *Journal of Physics*.

Farhin Mansur, Vibha Patel, Mihir Patel. (2017). A Review on Recommender System. *ICIIECS.*

Francesco Ricci, Lior Rokach and Bracha Shapira. (2011). Recommender Systems Handbook. In *Recommender Systems Handbook.* Springer Science+Business Media.

G. Adomavicius, A. Tuzhilin. (2005). Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *Knowledge and Data Engineering, IEEE Transactions, 17*.

Greg Linden,Brent Smith,and Jeremy York. (2003). *Amazon.com Recommendations: Item-to-Item Collaborative Filtering.* IEEE Computer Society.

Hegland, M. (2007). *The Apriori Algorithm Tutorial.*

J. L.,Herlocker, J. A.,Konstan, A.,Borchers, and J.,Riedl. (1999). An algorithmic framework for performing collaborative filtering. *ACM SIGIR conference on Research and development in information retrieval.*

Jiawei Han, Micheline Kamber, Jian Pei. (2012). *Data Mining: Concepts and Techniques.* Elsevier Inc.

Jiawei Han, Jian Pei, and Yiwen Yin. (n.d.). Mining Frequent Patterns without Candidate Generation.

JinHyun Jooa, SangWon Bangb, GeunDuk Parka. (2016). Implementation of a Recommendation System using Association Rules and Collaborative Filtering. *Information Technology and Quantitative Management (ITQM 2016).*

Karandeep Singh Talwar, Abhishek Oraganti, Ninad Mahajan, Pravin Narsale. (2015). Recommendation System using Apriori Algorithm. *IJSRD - International Journal for Scientific Research & Development, 3*(01).

Marko Balabanovic,Yoav Shoham. (1997). Fab: content-based, collaborative recommendation. *Communications of the ACM*.

N.,Bendakir, and E.,Aımeur. (2006). Using association rules for course recommendation. *AAAI Workshop on Educational Data Mining*, (pp. 31-40).

Nirmal Kaur, Gurpinder Singh*. (2017). A Review Paper On Data Mining And Big Data. *International Journal of Advanced Research in Computer Science, 8*(4).

R. Perego,P. Palmerini,S. Orlando. (n.d.). Enhancing the Apriori Algorithm for Frequent Set Counting. *Data Warehousing and Knowledge Discovery.*

R., B. (2002). Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction.*

Rakesh Agrawal, Ramakrishnan Srikant*. (1994). Fast Algorithms for Mining Association Rules. *VLDB Conference.* Santiago, Chile.

Ranjan S G, Sandesh A Hegde, Sujay N, Swaroop S Rao, Padmini M S. (2017). Improving Recommendation in E-Commerce Using Apriori Algorithm. *International Research Journal of Engineering and Technology (IRJET), 04*(04).

Riverbank. (2020, Jan 06). *Python bindings for the Qt cross platform application toolkit.* Retrieved from PYPI ORG: https://pypi.org/project/PyQt5/

S. Anand, B. (2006). Mobasher Intelligent Techniques for Web Personalization. *IJCAI*.

S.O. Abdulsalam, K.S. Adewole, A.G. Akintola, M.A. Hambali. (2014). Data Mining in Market Basket Transaction: AnAssociation Rule Mining Approach. *International Journal of Applied Information Systems (IJAIS), 7*(10).

Sagar Bhise, Prof. Sweta Kale. (2017). Effieient Algorithms to find Frequent Itemset Using Data Mining. *International Research Journal of Engineering and Technology, 04*(06).

Shadi AlZu'bi, Bilal Hawashin, Mohammad ElBes, Mahmoud Al-Ayyoub. (2018). A Novel Recommender System Based on Apriori Algorithm for Requirements Engineering. *2018 Fifth International Conference on Social Networks Analysis, Management and Security (SNAMS).*

T.,Chellatamilan, R.,SURESH. (2011). An e-Learning Recommendation System using Association Rule Mining Technique. *European Journal of Scientific Research*.

Wikipedia. (n.d.). *Apriori algorithm*. (Wikipedia) Retrieved 09 20, 2019, from https://en.wikipedia.org/wiki/Apriori_algorithm

X.,Su, and T. M.,Khoshgoftaar. (2009). A survey of collaborative filtering techniques. *Advances in Artificial Intelligence.*

Yabing, J. (2013). Research of an Improved Apriori Algorithm in Data Mining Association Rules. *International Journal of Computer and Communication Engineering, 2*(1).

Z., X. (2007). Building personalized recommendation system in E-commerce using association rule-based mining and classification. *Eighth ACIS International Conference* (pp. 853-857). IEEE.

Appendix A: Importing and Format Data

```
def import_data_from_sqlserver(source_table,source_columns,target_columns,t
arget_conn):
    try:
        connection = pymssql.connect(server='kim-
pc\sql2014', user='sa', password='blue123', database='orange_market')
        cursor = connection.cursor(as_dict=True)
        sqlserver = "select " + ",".join(source_columns) + " from [" + sour
ce_table + "] where posdate between '2017-01-01' and '2017-12-31'"
        cursor.execute(sqlserver)
        records = cursor.fetchall()
        values = []
        for row in records:
            row_values = []
            for column in (source_columns):
                row_values.append(row.get(str(column)))
            values.append(row_values)

        cursor.close()
        connection.close()
        # ====================
        query = "insert into sales_transaction (" + ",".join(target_columns
) + ") VALUES (" + ",".join(["%s"] * len(target_columns))+ ")"
        cursor = target_conn.cursor()
        cursor.execute("delete from sales_transaction")
        cursor.executemany(query,values)
        target_conn.commit()
        # preprocessing data
        sql = 'delete from preprocessing_transaction'
        cursor = target_conn.cursor()
        cursor.execute(sql)
        sql = "SELECT d.document_no, "
        sql = sql + " GROUP_CONCAT(DISTINCT d.item_no SEPARATOR ',') "
        sql = sql + " as item FROM sales_transaction as d "
        sql = sql + " GROUP BY d.document_no "
        insert_query = "insert into preprocessing_transaction"
        insert_query = insert_query + "(document_no,item) " + sql
        cursor.execute(insert_query)
        print(insert_query)
        target_conn.commit()
        cursor.close()
    except Exception as error:
        result = str(error)
if __name__ == "__main__":
    to_connection = create_open_database(host='localhost',port=3307,user='r
oot',password='blue123',db_name='ksapriori')

    source_table = 'POSDETAIL'
```

```
    source_columns = ['INVID', 'PRODUCTID']
    target_columns = ['document_no', 'item_no']
    import_data_from_sqlserver(source_table,source_columns,target_columns,t
o_connection)
```

Appendix B: Frequent itemset generation of the Apriori Algorithm

```
def ksapriori(dataset,min_support):
    # start from size 1
    c1 = list(create_c1(dataset))
    data = list(map(set, dataset))
    f1, support_data = scan_dataset(data, c1, min_support)
    #================================
    freq_itemsets = [f1]
    k = 2
    while(len(freq_itemsets[k-2]) > 0):
        # print ('Level : ', len(candidate_list[k-2]))
        ck =   create_ck(freq_itemsets[k-2], k)
        fk, support_data_k = scan_dataset(data, ck, min_support)
        support_data.update(support_data_k)
        freq_itemsets.append(fk)
        k += 1
    return freq_itemsets, support_data
```

Appendix C: Create Candidates Itemset

```
def create_c1(dataset):
    """
    Create a list of unique items in transaction data.
    Represent each item as a set of length 1.
    """
    c = []
    for data in dataset:
        for item in data:
            if not [item] in c:
                c.append([item])
    c.sort()
    return list(map(frozenset, c))

def create_ck(frequent_itemset, k):
    """
    Create a list of candidates of length k.
    Arguments:
        frequent_itemset: a list of frequent itemsets
        k: the size of the itemsets

    """
    candidate_list = []
```

```
    len_freq_itemsets = len(frequent_itemset)
    for i in range(len_freq_itemsets):
        for j in range(i + 1, len_freq_itemsets):
            L1 = list(frequent_itemset[i])[:k-2]
            L2 = list(frequent_itemset[j])[:k-2]
            L1.sort()
            L2.sort()
            if L1 == L2:
                candidate_list.append(frequent_itemset[i] | frequent_itemse
t[j])
    return candidate_list
```

Appendix D: Counting Support and remove if less than minimum support

```
def scan_dataset(data, candidates, min_support):
    """
    Scan through transaction data and return a list of candidates that meet
    the minimum support threshold, and support data about the current candi
dates.
    Arguments:
        data: data set,
        candidates: a list of candidate sets
        min_support: the minimum support
    """
    count = {}
    for tid in data:
        for candidate in candidates:
            if candidate.issubset(tid):
                if not candidate in count: count[candidate] = 1
                else: count[candidate] += 1
    num_of_trans = float(len(data))
    candidate_list = []
    support_data = {}
    # calculate support for every itemset
    for key in count:
        support_count = count[key]
        support = count[key] / num_of_trans # in percentage
        # If the support meets the minimum support requirements,
        # add it to the list of itemsets.
        if support >= min_support:
            candidate_list.insert(0,key)
        support_data[key] = support
    return candidate_list, support_data
```

Appendix E: Association rule generation

```
def calculate_confidence(freqent_itemset,next_itemset,support_data,min_conf
idence,rule_list):
    """
    Arguments:
        frequent_itemsets: a list of frequent_itemset
```

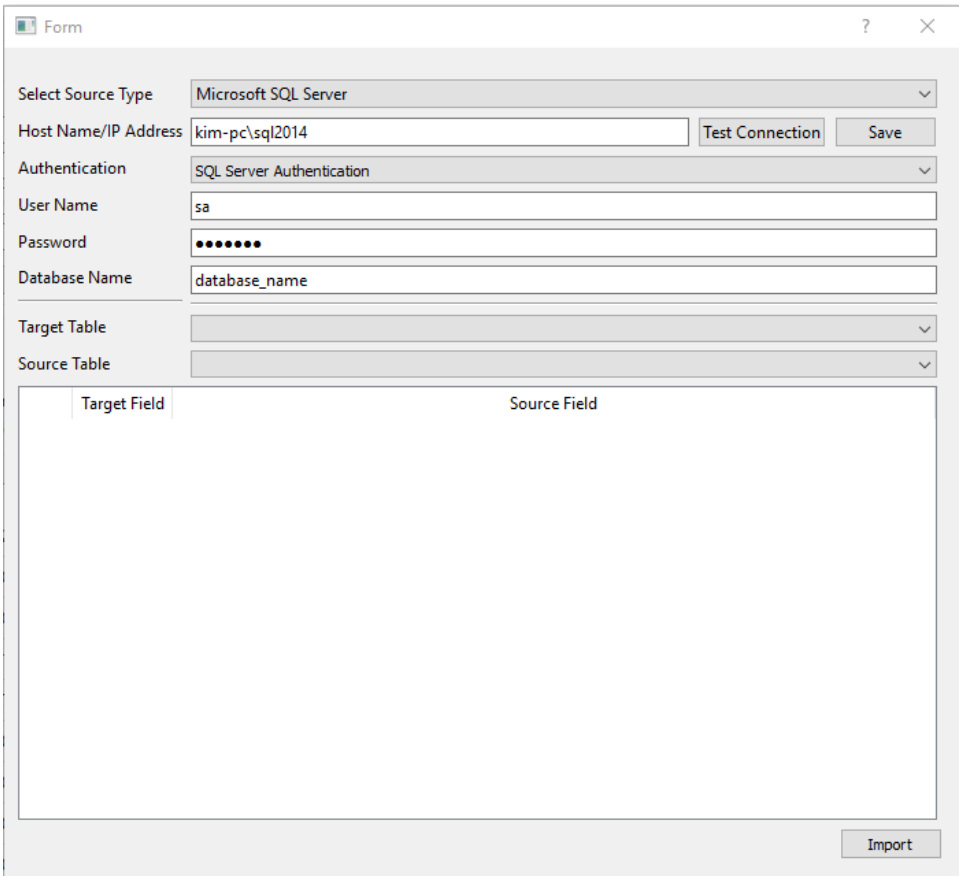
```
        next_itemset : a list of next iteration
        support_data: a list of itemsets support data
        min_confidence: a minimum confidence threshold in percentage
        rule_list : a list of association rules

        Return as Pruned List
    """
    pruned_list = []
    for consequent in next_itemset:
        confidence = support_data[freqent_itemset] / support_data[freqent_i
temset - consequent]
        if confidence >= min_confidence:
            rule_list.append((freqent_itemset - consequent, consequent, sup
port_data[freqent_itemset], confidence))
            pruned_list.append(consequent)
    return pruned_list
```

```
def rules_from_consequent(freqent_itemset,next_itemset,support_data,min_con
fidence,rule_list):
    """
    Arguments:
        frequent_itemsets: a list of frequent_itemset
        next_itemset : a list of next iteration
        support_data: a list of itemsets support data
        rule_list : a list of association rules

        Return as Pruned List
    """
    tmp1 = []
    m = len(next_itemset[0])
    if (len(freqent_itemset) > (m + 1)):
        tmp1 = create_ck(next_itemset, m + 1) # Gen list of next iteration
        tmp1 = calculate_confidence(freqent_itemset, tmp1, support_data,min
_confidence, rule_list) # pruning. pick qualified rules.
    if (len(tmp1) > 1):
        calculate_confidence(freqent_itemset, tmp1, support_data,min_confid
ence, rule_list) # Continue\Iterate to next level
```

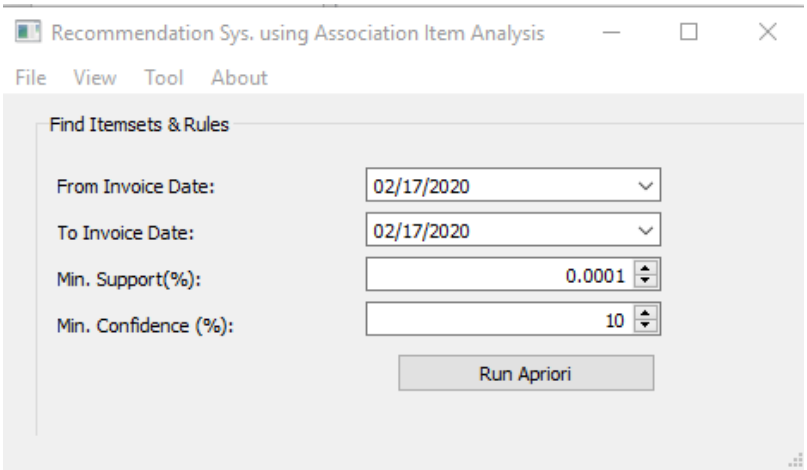
```
def generate_rule(frequent_itemsets,support_data,min_confidence):
    rule_list = []
    min_confidence = min_confidence / 100
    try:
        for i in range(1, len(frequent_itemsets)):
            for freq_itemset in frequent_itemsets[i]:
                # {0,1,2} -> [{0},{1},{2}].
                next_itemset = [frozenset([item]) for item in freq_itemset]
                if (i > 1): # length > 2, go level by level
```

```
                    rules_from_consequent(freq_itemset,next_itemset,support
_data,min_confidence,rule_list)
                else: # if only 2 items, just prune - the base
                    calculate_confidence(freq_itemset,next_itemset,support_
data,min_confidence,rule_list)
    except Exception as error:
        print("Rule : " + str(error))

    return rule_list
```

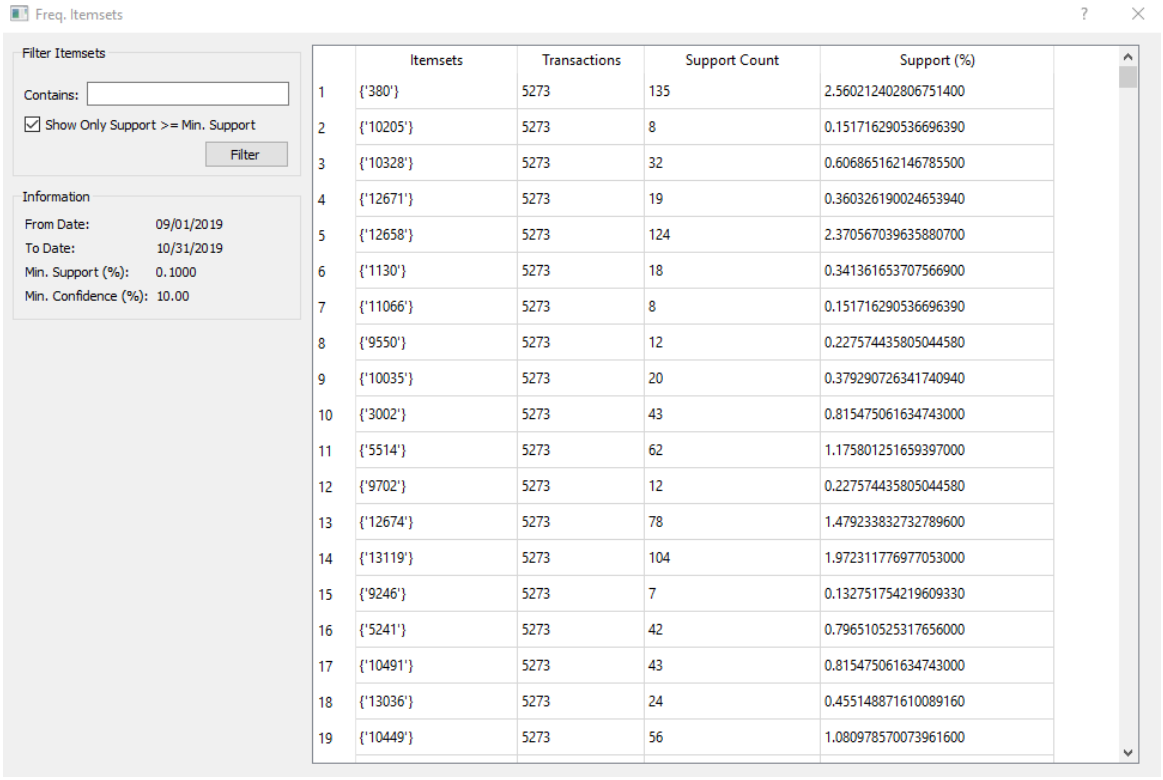
Appendix F: Graphical user interface for importing data



Appendix G: GUI for Main Application

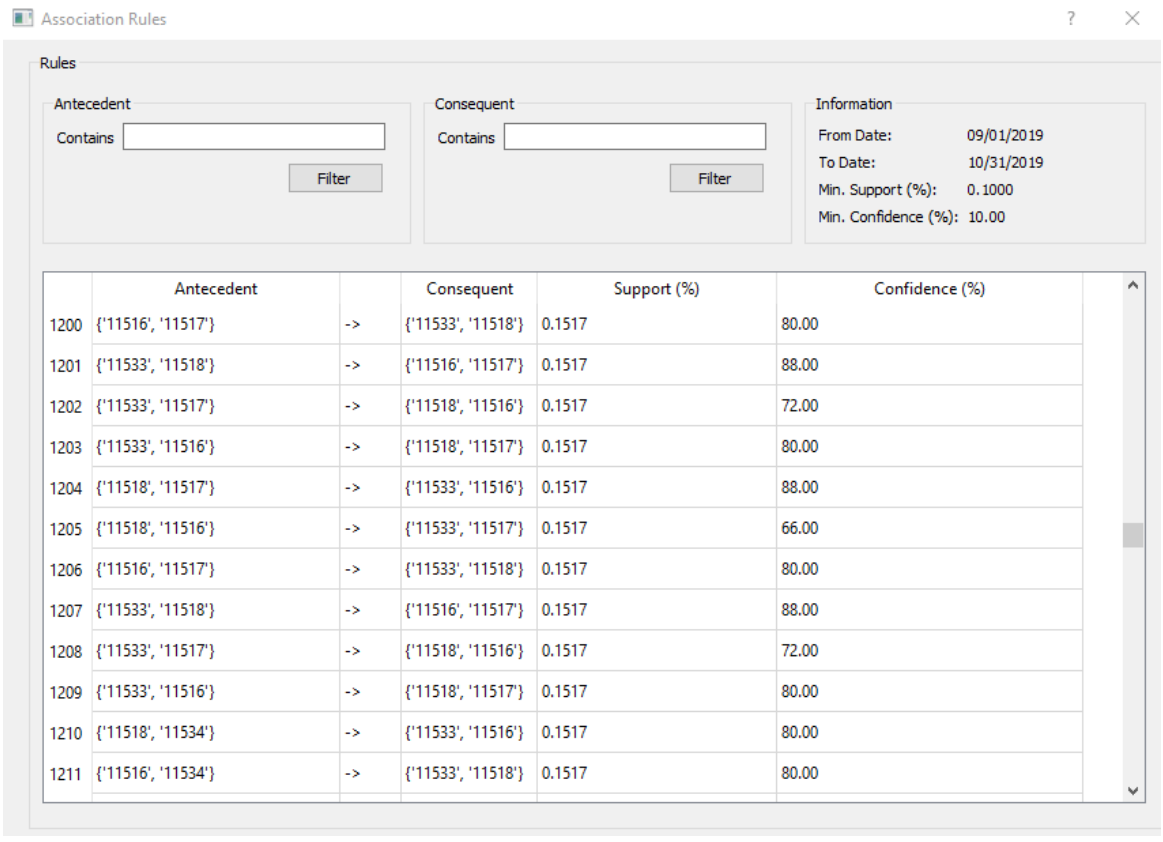


Appendix H: GUI for frequent itemset generation of the Apriori Algorithm



	Itemsets	Transactions	Support Count	Support (%)
1	{'380'}	5273	135	2.560212402806751400
2	{'10205'}	5273	8	0.151716290536696390
3	{'10328'}	5273	32	0.606865162146785500
4	{'12671'}	5273	19	0.360326190024653940
5	{'12658'}	5273	124	2.370567039635880700
6	{'1130'}	5273	18	0.341361653707566900
7	{'11066'}	5273	8	0.151716290536696390
8	{'9550'}	5273	12	0.227574435805044580
9	{'10035'}	5273	20	0.379290726341740940
10	{'3002'}	5273	43	0.815475061634743000
11	{'5514'}	5273	62	1.175801251659397000
12	{'9702'}	5273	12	0.227574435805044580
13	{'12674'}	5273	78	1.479233832732789600
14	{'13119'}	5273	104	1.972311776977053000
15	{'9246'}	5273	7	0.132751754219609330
16	{'5241'}	5273	42	0.796510525317656000
17	{'10491'}	5273	43	0.815475061634743000
18	{'13036'}	5273	24	0.455148871610089160
19	{'10449'}	5273	56	1.080978570073961600

Appendix I: GUI for association rules



	Antecedent		Consequent	Support (%)	Confidence (%)
1200	{'11516', '11517'}	->	{'11533', '11518'}	0.1517	80.00
1201	{'11533', '11518'}	->	{'11516', '11517'}	0.1517	88.00
1202	{'11533', '11517'}	->	{'11518', '11516'}	0.1517	72.00
1203	{'11533', '11516'}	->	{'11518', '11517'}	0.1517	80.00
1204	{'11518', '11517'}	->	{'11533', '11516'}	0.1517	88.00
1205	{'11518', '11516'}	->	{'11533', '11517'}	0.1517	66.00
1206	{'11516', '11517'}	->	{'11533', '11518'}	0.1517	80.00
1207	{'11533', '11518'}	->	{'11516', '11517'}	0.1517	88.00
1208	{'11533', '11517'}	->	{'11518', '11516'}	0.1517	72.00
1209	{'11533', '11516'}	->	{'11518', '11517'}	0.1517	80.00
1210	{'11518', '11534'}	->	{'11533', '11516'}	0.1517	80.00
1211	{'11516', '11534'}	->	{'11533', '11518'}	0.1517	80.00