Clustering Based Approach to Enhance Association Rule Mining

MSc Research Project
MSc Data Analytics

Samruddhi Shailesh Kanhere
Student ID: x18190634

School of Computing
National College of Ireland

Supervisor: Dr. Paul Stynes, Dr. Pramod Pathak

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Samruddhi Shailesh Kanhere
Student ID:	x18190634
Programme:	MSc Data Analytics
Year:	2020
Module:	MSc Research Project
Supervisor:	Dr. Paul Stynes, Dr. Pramod Pathak
Submission Due Date:	28/09/2020
Project Title:	Clustering Based Approach to Enhance Association Rule Mining
Word Count:	1048
Page Count:	10

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Samruddhi Shailesh Kanhere
Date:	28th September 2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Clustering Based Approach to Enhance Association Rule Mining

Samruddhi Shailesh Kanhere
x18190634

1 Introduction

The configuration manual is written so that the research work can be reproduced. The topic of my research is **"Clustering based approach to enhance association rule mining"**. It includes all the details about the System Setup i.e. hardware and software requirements, programming languages and libraries that are used. Further, this manual contains all the steps to run the code. In the last section, the important code snippets are attached.

2 System Setup

This section has the steps which will help in setting up the environment. The hardware and software requirements are listed below.

2.1 Hardware Requirements

The research is performed on a personal laptop. Below are the configurations of the same.

- Operating System: Windows OS, 64-bit
- Storage: 1 TB HDD
- RAM: 8.00 GB
- Processor: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz

2.2 Software Requirements

This section lists the softwares that needs to be installed before running the code.

- **Microsoft Excel 2016**

  Excel is a widely used spreadsheet tool developed by Microsoft. This is used to store the data is the Comma Separated Values (CSV) format. This tool is also used for capturing the results of the experiments.

- **RStudio Desktop Version 1.2.1335**

  RStudio is an open source software. It provides an Integrated Development Environment for R. It can be downloaded from RStudio official website[1]. Before installing RStudio, R needs to be installed. R is an open source programming language specifically designed for statistical computing. It can be downloaded from CRAN website[2]. The R version used in this research is 3.6.0. It is used for exploratory data analysis, data preparation, data modelling, etc.

- **Anaconda Navigator Distribution: Spyder 3.3.6**

  Anaconda Navigator is an open source platform for Python operations. Python version 3.7 is used in this research. Python is used for Data modelling in this research. Anaconda can be downloaded from the official website of Anaconda[3]. Python can be downloaded from the Python's official website [4].

- **Notepad ++ 7.7.1**

  Notepad ++ is an open source editor. This is used for scratch work in this research. It can be downloaded from Notepad official website[5].

3 Libraries

3.1 Python libraries

Python is mainly used for implementation of Frequent Pattern (FP) Growth algorithm. The Python libraries listed in Table 1 are used in this research. To install Python packages, use the following command in the anaconda prompt[6].
conda install package_name

Table 1: Python libraries

Library	Version	Purpose
datetime	0.4-3	record the execution times
json	0.8.5	dealing with Json data
matplotlib	3.1.1	visualize the results
pandas	0.25.1	reading data
pyfpgrowth	1.0	implement FP Growth
seaborn	0.9.0	creating visualizations

[1] https://rstudio.com/products/rstudio/download/
[2] https://cran.r-project.org/bin/windows/
[3] https://www.anaconda.com/products/individual
[4] https://www.python.org/downloads/
[5] https://notepad-plus-plus.org/downloads/v7.7.1/
[6] https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-pkgs.html

3.2 R libraries

The R libraries listed in Table 2 are used in this research. To install these R packages, following command should be executed[7].
install.packages("package name")

Table 2: R libraries

Library	Version	Purpose
arules	1.6-6	implement association rule mining
arulesViz	1.3-3	visualize the association rules and frequent itemsets
clValid	0.6-9	validate the clustering results
clustree	0.4-3	visualizing the clusters
dplyr	1.0.0	data manipulation
factoextra	1.0.7	find the optimal number of clusters
FactoMineR	2.3	exploratory data aanlysis
ggfortify	0.4.10	Visualization of statistical results
GGally	2.0.0	creating plots
ggiraphExtra	0.2.9	making interactive plots
jsonlite	1.6	json parser
lubridate	1.7.4	date operations
NbClust	3.0	implementing clustering
plyr	1.8.4	tools to split and combine data
RColorBrewer	1.1-2	colour brewer palettes
tidyverse	1.2.1	text data manipulation

4 Code Snippets

This section includes the important code snippets from the overall project development process. The important steps in this process are data preparation, data modelling, and evaluation. Before that, Table 3 shows the important columns from the dataset and its description.

Table 3: Dataset Description

Column	Type	Description
RequestBasketValue	Decimal	Full basket value as per the bill
ResponseProcessingTimestamp	Date Time (UTC)	dealing with Json data
RequestBasketId	Varchar	Transaction id
RequestNumberBasketItems	Integer	Number of items in transaction
ResponseFinancialTimestamp	Date Time (UTC)	Transaction processing time
RequestBasketJsonString	Varchar	JSON string containing basket items

[7] http://jtleek.com/modules/01_DataScientistToolbox/02_09_installingRPackages/#11

4.1 Data Preparation

The first step in the project development process is data preparation. It includes all the operations related to data importing, data pre-processing and data transformation. These operations are performed in the RStudio.

```r
###############################Importing data##########################################################
#Load the data files into the R environment
BigTempRb <- read.csv("E:/NCI/SEM3/RB/20181129_PMRB_0001.txt")
Products <- read.csv("E:/NCI/SEM2/RIC/RetailData/Promopay/tcg_products_20181219.csv")

#############################################Data Pre-processing####################################
#Converting the JSON data into the structured format.
json <- BigTempRb$RequestBasketJsonString[1:300000] #Only taking 300,000 records due to system limitations
nested_dataframe <- jsonlite::stream_in(textConnection(gsub("\\n", "", json)))
nested_dataframe <- cbind(nested_dataframe, dt=BigTempRb$ResponseFinancialTimestamp[1:300000], basket_items=BigTempRb$RequestNumberBasketItems[1:300000])
df <- nested_dataframe %>% unnest(items)
df = df[-c(2)]     # Remove the unwanted column Curr
names(Products)[1] <- "b" # Rename Product table EAN to b
df <- inner_join(df, Products, by.="b")
df1 <- df

#Removing the items having negative quantity. Also, the baskets with single items are ignored.
df <- filter(df, b != '5000128785617')
df <- filter(df, b != '0000000000145')
df <- filter(df, basket_items != "1")
df <- filter(df, q >= 1)

head(df)
###################### Changing Date Time format to Categorical ####################################
#For differential MBA, we need to change the time of transaction into a categorical variable.
#The transactions are classified into 4 classes: Morning, Afternoon, Evening, Night
library(lubridate)
library(plyr)
test1 <-   mapvalues(hour(df$dt),from=c(0:23),
                     to=c(rep("Night",times=5), rep("Morning",times=6),rep("Afternoon",times=5),rep("Evening", times=4) ,rep("Night", times=4)))
df <- cbind(df, time_cat=test1)
head(df)
```

Figure 1: Data Preparation

As shown in Figure 1, the data pre-processing includes the conversion of semi-structured JSON data into structured format, handling missing values and inconsistencies, changing the date format, etc. This is how data is prepared for further modelling.

4.2 Data Modelling

Data modelling section is all about the implementation of the research. This section includes experiment wise code snippets.

- **Experiment 1: Replication of state-of-the-art (Hossain et al.; 2019).**

  Two datasets are used for this experiment. These datasets are downloaded from Kaggle. This experiment is performed in Python language.

  - **Dataset 1[8]: French Retail Dataset**

```python

# reading the dataset
data = pd.read_csv('E:\\NCI\\SEM3\\Thesis\\Code\\Dataset\\StateOfArt\\Market_Basket_Optimisation.csv', header = None)

#Data Pre-processing
transactions = []
for i in range(0, 7501):
    transactions.append([str(data.values[i,j]) for j in range(0, 20) if str(data.values[i,j]) != 'nan'])

```

Figure 2: French Retail Dataset Preprocessing

  Figure 2,3,4 shows the code snippets for french retail dataset.

  - **Dataset 2[9]: Bakery dataset**

```python

# let's check the shape of the dataset
data.shape
plt.rcParams['figure.figsize'] = (15, 15)
wordcloud = WordCloud(background_color = 'white', width = 1200,  height = 1200, max_words = 121).generate(str(data[0]))
plt.imshow(wordcloud)
plt.axis('off')
plt.title('Most Popular Items',fontsize = 20)
plt.show()

# looking at the frequency of most popular items
plt.rcParams['figure.figsize'] = (18, 7)
color = plt.cm.copper(np.linspace(0, 1, 40))
data[0].value_counts().head(40).plot.bar(color = color)
plt.title('frequency of most popular items', fontsize = 20)
plt.xticks(rotation = 90 )
plt.grid()
plt.show()

y = data[0].value_counts().head(50).to_frame()
y.index

# plotting a tree map
plt.rcParams['figure.figsize'] = (20, 20)
color = plt.cm.cool(np.linspace(0, 1, 50))
squarify.plot(sizes = y.values, label = y.index, alpha=.8, color = color)
plt.title('Tree Map for Popular Items')
plt.axis('off')
plt.show()
```

Figure 3: French Retail Dataset: Exploratory Data Analysis

```python

#Define threshold values for FP Growth algorithm
min_support = 0.01
min_confidence = 0.5
support = round(len(transactions)*min_support)

#Generating Freqent Itemsets
patterns = pyfpgrowth.find_frequent_patterns(transactions, support)

#Generating Association Rules
rules = pyfpgrowth.generate_association_rules(patterns, min_confidence)

```

Figure 4: French Retail Dataset: FP Growth Implementation

```python
#Read the data
bakery_df = pd.read_csv('E:\\NCI\\SEM3\\Thesis\\Code\\Dataset\\StateOfArt\\BakeryDataset.csv')
bakery_df.head()
bakery_df.info()

#Pre-processing the data
#Converting items to lower abd removing missing values.
bakery_df['Item'] = bakery_df['Item'].str.lower()
x = bakery_df['Item'] == "none"
print(x.value_counts())
bakery_df = bakery_df.drop(bakery_df[bakery_df.Item == 'none'].index)
len(bakery_df['Item'].unique())

```

Figure 5: Bakery Dataset Preprocessing

Figure 5,6,7 shows the code snippets for bakery dataset.

This experiment is performed several times and execution times are captured by changing the support values.

[8]https://www.kaggle.com/roshansharma/market-basket-optimization
[9]https://www.kaggle.com/sulmansarwar/transactions-from-a-bakery

```

#Top 20 top selling items
fig, ax=plt.subplots(figsize=(16,7))
bakery_df['Item'].value_counts().sort_values(ascending=False).head(20).plot.bar(width=0.5,edgecolor='k',align='center',linewidth=1)
plt.xlabel('Food Item',fontsize=20)
plt.ylabel('Number of transactions',fontsize=17)
ax.tick_params(labelsize=20)
plt.title('20 Most Sold Items at the Bakery',fontsize=20)
plt.grid()
plt.ioff()

```

Figure 6: Bakery Dataset: Exploratory Data Analysis

```
#Trasnforming data into required format
grouped_df = bakery_df.groupby('Transaction')
grouped_lists = grouped_df["Item"].apply(list)
grouped_lists = grouped_lists.reset_index()

print(len(grouped_lists))
#Define threshold values
min_support = 0.01
min_confidence = 0.5
support = round(len(grouped_lists)*min_support)

#Calculating Frequent Itemsets
patterns = pyfpgrowth.find_frequent_patterns(grouped_lists["Item"], support)
#Generating Association Rules
rules = pyfpgrowth.generate_association_rules(patterns, min_confidence)
print(rules)

```

Figure 7: Bakery Dataset: FP Growth Implementation

- **Experiment 2: Implementing FP Growth for Glantus dataset**.

  The FP Growth algorithm is implemented on Glantus dataset by keeping the same parameters as state-of-the-art (Hossain et al.; 2019). This experiment is performed in Python. Figure 8, 9 show the code snippet for Experiment 2.

```
## Reading the transactions file
df=pd.read_csv("E:\\NCI\\SEM2\\RIC\\Code\\Dataset\\BigTempRb.txt",delimiter=',')

x = datetime.datetime.now()
print("Program beginning at :",x)
df1 = pd.DataFrame(columns=['ResponseRgBasketId', 'BasketID', 'ItemID','Qty','Price'])
## json parser for parsing the json basket strings
itemlist=[]
for i in range(0,len(df[["ResponseRgBasketId"]])):
    response=json.loads(df["RequestBasketJsonString"].iloc[i])
    #for nest in response['items']:
    lister = []
    for nest in response['items']:
        lister.append(nest["b"])
    itemlist.append(list(dict.fromkeys(lister)))
```

Figure 8: Glantus Dataset: Data Preprocessing

- **Experiment 3: Performance Comparison of Apriori, FP Growth, and**

```
#Set same parameters as state-of-the-art
minimum_sup = 0.1
minimum_conf = 0.5
#FP Growth algorithm to generate the rules.
support = round(len(itemlist)*minimum_sup)
#Generate frequent patterns and association rules.
patterns = pyfpgrowth.find_frequent_patterns(itemlist, support)
rules = pyfpgrowth.generate_association_rules(patterns, 0.5)
print(rules)
```

Figure 9: Glantus Dataset: FP Growth Implementation

Eclat algorithms.

Glantus dataset is used for this experiment. These three association rule mining algorithms are implemented. The experiment is performed several times by to record the execution times by changing the number of input transactions in each iteration. The readings are recorded in Microsoft Excel.

- **Apriori Algorithm**

  Apriori algorithm is implemented on the Glantus dataset after completing the pre-processing explained in Section 4.1. This is implemented in R language. Figure 10 shows the code snippet for the same.

```
#Track Time#### Apriori Pre-processing################################################################
algo_start_time = Sys.time()
basket <- as(Transactions_by_Subcategory, "transactions")
itemFrequencyPlot(basket,topN=20,type="absolute",col=brewer.pal(8,'Pastel2'), main="Absolute Item Frequency Plot")
#itemFrequencyPlot(basket,topN=20,type="relative",col=brewer.pal(8,'Pastel2'),main="Relative Item Frequency Plot")
summary(basket)
######## Apriori ###########################################################################
association_rules <- apriori(basket, parameter = list(supp=0.002, conf=0.005,maxlen=10,target="rules",minlen=2))
summary(association_rules) #shows the following:
rules <- association_rules[!is.redundant(association_rules)]
as2221 <- inspectDT(sort(association_rules, by = "confidence"))
inspectDT(sort(association_rules, by = "confidence"))
algo_end_time = Sys.time()
algo_time = difftime(algo_end_time,algo_start_time,units = "secs")
```

Figure 10: Glantus Dataset: Apriori Algorithm

- **Eclat Algorithm**

  In this section, Eclat algorithm is implemented using R language. Before doing that, data pre-processing is performed. Figure 11 shows the code snippet for the same.

```
######## ECLAT Algorithm###################################################################################
eclat_frequent_itemsets = eclat(data = basket, parameter = list(support = 0.002, minlen = 2 ))
eclat_rules <- ruleInduction(eclat_frequent_itemsets, basket, confidence = 0.005)
inspectDT(sort(eclat_rules, by = 'support'))
```

Figure 11: Glantus Dataset: Eclat Algorithm

- **FP Growth Algorithm**

FP Growth algorithm is implemeted using Python language. The values of input parameters are changed than the previous experiment. Figure 12 shows the code snippet for the same.

```
## Reading the transactions file
my_df=pd.read_csv("E:\\NCI\\SEM3\\RB\\20181129_PMRB_0001.txt", delimiter=',')
df = my_df[1:100000]
x = datetime.datetime.now()
print("Program beginning at :",x)
df1 = pd.DataFrame(columns=['ResponseRgBasketId', 'BasketID', 'ItemID','Qty','Price'])

## json parser for parsing the json basket strings
itemlist=[]
for i in range(0,len(df[["ResponseRgBasketId"]])):
    response=json.loads(df["RequestBasketJsonString"].iloc[i])
    #for nest in response['items']:
    lister = []
    for nest in response['items']:
        lister.append(nest["b"])
    itemlist.append(list(dict.fromkeys(lister)))

#Setting Parameters
minimum_sup = 0.002
minimum_conf = 0.005
#FP Growth algorithm to generate the rules.
support = round(len(itemlist)*minimum_sup)
patterns = pyfpgrowth.find_frequent_patterns(itemlist, support)
rules = pyfpgrowth.generate_association_rules(patterns, minimum_conf)
print(rules)
y = datetime.datetime.now()
print("Program ending at :",y)
print("total time : ", (y-x).total_seconds())
```

Figure 12: Glantus Dataset: FP Growth Algorithm

- **Experiment 4: Implement Clustering based approach for dataset reduction in Association Rule Mining**.

  In this experiment, K-means clustering is implemented and based on the results of K-means the dataset is reduced. After dataset reduction, the association rule mining algorithms are implemented. The Figure 13 and Figure 14 show the code snippets for K-means clustering. The frequency and average price is calculated for all the products as a part of preprocessing. Then, K-means clustering is implemented based on these calculated parameters. The implementation is done in R. Figure 15 shows the snippet to filter the data based on clustering results.

- **Experiment 5: Implementing differential market basket analysis for Glantus dataset**.

  Transactions are groups based on the time. The four groups are 'Afternoon', 'Morning, 'Night', and 'Evening'. Then, the association rule mining algorithms are implemented on each group and the output is compared. The implementation is performed in R. Figure 16 and Figure 17 show the code snippets for this experiment.

```
######################### K Means Clustering ####################################################################
Frequency_Table <- count(df, 'b')
library(dplyr)
#max_length_trans <- df %>% count(id)
#View(max_length_trans)
#view(Frequency_Table)
Frequency_Table$b[Frequency_Table$b == ""] <- NA
Frequency_Table <- na.omit(Frequency_Table, na.string = "")

test2 <-  cbind(df$b, df$p)
test2 <- as.data.frame(test2)
test2[test2 == ""] <- NA
test2 <- na.omit(test2, na.string = "")

#Find the average price
test2$V2 <- as.numeric(as.character(test2$V2))
test2 <- aggregate(.~V1, data=test2, mean)
#Rename the columns
names(test2)[1] <- "b"
names(test2)[2] <- "price"
Frequency_Table <- inner_join(Frequency_Table, test2, by="b")
```

Figure 13: Glantus Dataset: Clustering Pre-processing

```
#Find the optimum numbers of clusters
fviz_nbclust(Frequency_Table, kmeans, method = "silhouette", k.max = 24) + theme_minimal() + ggtitle("The Silhouette Plot")

## Optimum number of clusters is 4 after running "silhouette" Clustering for the data
set.seed(0)
km.res <- kmeans(Frequency_Table[,2:3], 4)

km.res$cluster
km.res$size
km.res$centers

# vary parameters for most readable graph
clusplot(Frequency_Table, km.res$cluster, color=TRUE, shade=TRUE,
         labels=2, lines=0)
```

Figure 14: Glantus Dataset: Clustering Implementation

```
##### Mapping cluster number to each transaction ######
cl_df <- inner_join(Frequency_Table[,c(1,4)], df, by="b")
cl_df$time_cat <- as.character(cl_df$time_cat)
cl_trans <- cl_df
#view(cl_trans)
cl_trans <- filter(cl_trans, cluster != '2')
a <- cl_trans
```

Figure 15: Filtering data based on Clustering

```
######Clustering the data based on the time of the day
df_morning <- cl_df %>% filter (cl_df$time_cat=='Morning')
df_afternoon <- cl_df %>% filter (cl_df$time_cat=='Afternoon')
df_evening <- cl_df %>% filter (cl_df$time_cat=='Evening')
df_night <- cl_df %>% filter (cl_df$time_cat=='Night')
```

Figure 16: Grouping data based on time

Figure 17 shows the Apriori algorithm implemented on Morning transactions. Similarly, it is implemented on the remaining groups.

```
###Calculating rules for morning time.###################################################################
df_morning$id <- as.factor(df_morning$id)
Morning_Transactions_by_Subcategory <- split(df_morning$b, df_morning$id)
#preproc_end_time = Sys.time()
#preproc_time = difftime(preproc_end_time,preproc_start_time, units = "secs")

#Track Time#### Apriori Pre-processing
algo_start_time = Sys.time()
morning_basket <- as(Morning_Transactions_by_Subcategory, "transactions")
summary(morning_basket)
######## Apriori #######
morning_association_rules <- apriori(morning_basket, parameter = list(supp=0.002, conf=0.005,maxlen=10,target="rules",minlen=2))
summary(morning_association_rules) #shows the following:
morning_association_rules <- morning_association_rules[!is.redundant(morning_association_rules)]
inspectDT(sort(morning_association_rules, by = "confidence"))
algo_end_time = Sys.time()
algo_time = difftime(algo_end_time,algo_start_time,units = "secs")
```

Figure 17: Apriori on Morning group

References

Hossain, M., Sattar, S. A. H. M. and Paul, M. K. (2019). Market Basket Analysis Using Apriori and FP Growth Algorithm, *2019 22nd International Conference on Computer and Information Technology (ICCIT)* pp. 1–6.