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

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.

<table>
<thead>
<tr>
<th><strong>Signature:</strong></th>
<th>Samruddhi Shailesh Kanhere</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Date:</strong></td>
<td>28th September 2020</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th><strong>Office Use Only</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Signature:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Date:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Penalty Applied (if applicable):</strong></td>
<td></td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Library</th>
<th>Version</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>datetime</td>
<td>0.4-3</td>
<td>record the execution times</td>
</tr>
<tr>
<td>json</td>
<td>0.8.5</td>
<td>dealing with Json data</td>
</tr>
<tr>
<td>matplotlib</td>
<td>3.1.1</td>
<td>visualize the results</td>
</tr>
<tr>
<td>pandas</td>
<td>0.25.1</td>
<td>reading data</td>
</tr>
<tr>
<td>pyfpgrowth</td>
<td>1.0</td>
<td>implement FP Growth</td>
</tr>
<tr>
<td>seaborn</td>
<td>0.9.0</td>
<td>creating visualizations</td>
</tr>
</tbody>
</table>

[^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/
3.2 R libraries

The R libraries listed in Table 2 are used in this research. To install these R packages, the following command should be executed:

install.packages("package name")

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

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

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

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**: French Retail Dataset

    Figure 2: French Retail Dataset Preprocessing

    Figure 2 shows the code snippets for french retail dataset.

  - **Dataset 2**: Bakery dataset
# Figure 3: French Retail Dataset: Exploratory Data Analysis

```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'] = (15, 15)
color = plt.cm.copper(np.linspace(0, 1, 49))
data[0].value_counts().head(50).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))
plt.title('Tree Map for Popular Items')
plt.axis('off')
plt.show()
```

---

# Figure 4: French Retail Dataset: FP Growth Implementation

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

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

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

---

# Figure 5: Bakery Dataset Preprocessing

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

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

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.

https://www.kaggle.com/roshansharma/market-basket-optimization
https://www.kaggle.com/sulmansarwar/transactions-from-a-bakery
• 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 \cite{Hossain_2019}. This experiment is performed in Python. Figure 8 and 9 show the code snippet for Experiment 2.

```python
# Transforming 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 3: Performance Comparison of Apriori, FP Growth, and

```python
# Reading the transactions file
df=pd.read_csv("E:\\N\HIC\\SEH2\\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

for i in range(0,len(df["ResponseRgBasketId"])):   
    response=json.loads(df["ResponseBasketJsonString"][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
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.

- 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.

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

```python
# Reading the transactions file
my_df = pd.read_csv("E:\\NTI\\SEM3\\RB\\20181129_PM\R_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.
Figure 13: Glantus Dataset: Clustering Pre-processing

```
95. # K Means Clustering
96. Frequency_Table <- count(df, 'b')
97. library(DTyr)
98. max_length_trans <- df %>% count(id)
99. Avw(max_length_trans)
100. Avw(Frequency_Table)
101. Frequency_Table$[Frequency_Table == ''] <- NA
102. Frequency_Table <- na.omit(Frequency_Table, na.string = '')
103. test2 <- cbind(dfib, dfsp)
104. test2 <- as.data.frame(test2)
107. # Find the average price
108. test2$v2 <- as.numeric(as.character(test2$v2))
109. test2 <- aggregate(-v1, data=test2, mean)
110. # Rename the columns
111. names(test2)[1] <- 'b'
112. names(test2)[2] <- 'price'
113. Frequency_Table <- inner_join(Frequency_Table, test2, by="b")
```

Figure 14: Glantus Dataset: Clustering Implementation

```
114. # Find the optimum numbers of clusters
115. fviz_nbclust(Frequency_Table, kmeans, method = "silhouette", k.max = 24) + theme_minimal() + ggtitle("The Silhouette Plot")
116. # Optimum number of clusters is 4 after running "silhouette" clustering for the data
117. set.seed(0)
118. km.res <- kmeans(Frequency_Table[,2:3], 4)
119. km.res$cluster
120. km.res$size
121. km.res$centers
122. # Vary parameters for most readable graph
123. clusplot(Frequency_Table, km.res$cluster, color=TRUE, shade=TRUE, labels=2, times=0)
```

Figure 15: Filtering data based on Clustering

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

Figure 16: Grouping data based on time

```
143. # Clustering the data based on the time of the day
144. df_morning <- cl df %>% filter(cl df$time_cat=='Morning')
145. df_afternoon <- cl df %>% filter(cl df$time_cat=='Afternoon')
146. df_evening <- cl df %>% filter(cl df$time_cat=='Evening')
147. df_night <- cl df %>% filter(cl df$time_cat=='Night')
```

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

References