Association Rules and Item-Based Collaborative Filtering for Amazon.com Baseline Recommender System

Independent Study Thesis

Presented in Partial Fulfillment of the Requirements for
the Degree Bachelor of Arts in Statistical & Data Sciences in
the
Department of Mathematical & Computational Sciences at
The College of Wooster

by
Vinh Nghiem To
The College of Wooster
2022

Advised by:

Prof. Sofia Visa (Computer Science)

THE COLLEGE OF
WOOSTER

Abstract

Recommendation algorithms are best known for their use on e-commerce Web sites [24], where they use input about a customer's interests to generate a list of recommended items. In this study, the items that customers purchase and explicitly rate to represent their interests are applied into the recommendation models to personalize the online store for each customer. The dataset used for building the models is extracted from Amazon Customer Reviews, which contains customers' ratings in over two decades since the first review in 1995. We apply to a subset of this large dataset the association rule technique and the item-based collaborative filtering to identify additional items that a given customer might like to buy. Moreover, we discuss theoretically sentiment classification and analysis methods that can be useful in future work.

This work is dedicated to the future generations of Wooster students.

Acknowledgments

I would like to deeply thank my advisor, Dr. Visa, for her constant support and patience in guiding me through my independent study. To my friends at Wooster, especially Quan, Thinh, and Vy, and alumni such as Thuy and An, thank you for being a part of my memorable college experience. The meals that Thinh cooked have helped me a lot with my homesickness. It is also fun to procrastinate doing projects and reviewing exams until the last night they are due with Quan, you are really a 'lazy' friend that I've wanted to have. I was looking forward to playing PUBG and talking about numerous things with Thuy and An during weekends, and we usually laughed about how badly we played. I would also like to acknowledge my girlfriend, Tuyet Ngan, for all her mental support during my hardest time of my thesis. You are the luck of my life, my wonderwall and panacea. My warmest thank to my dad, who has always encouraged me no matter what I want to do and whom I learned a lot of precious life experience from.

Vita

Publications

Fields of Study Major field: Statistical & Data Sciences

Minor field: Economics and Math

Specialization: Applied Data Science in E-Commerce

Contents

List of Figures

List of Tables

List of Listings

Listing Page

CHAPTER 1

Introduction

1.1 Background

The ever increasing number of Internet users creates the need of a system able to provide users with assistance in searching for information and products based on their preferences and needs. Such search system, known as recommender systems, return to users meaningful results after surfing through massive amounts of, sometimes, disorganized data. Consequently, recommender systems are generating quite a lot of interest in certain academic fields and industry [11]. From the very beginning of the rapid development of computers and technology, large companies like Amazon have lead the race of implementing innovative recommender systems into the structure of the company to increase their competitiveness. Therefore, this paper would pay attention to application of recommender systems into e-commerce, which would be discussed more in the following sections.

All of us today have encountered recommendation systems in some facet of our daily lives as the next examples illustrate. Netflix suggests movies we can watch based on those we have already watched. Pandora deploys user feedback of liking or disliking a song to play music that we like. Facebook suggests new friends to add considering our current friends and their friends. YouTube recommends videos

similar to the ones we are currently viewing. An online store offers products or services based on our purchase history. These are just a few of the examples that might sound familiar. As data gets bigger, it is even more significant to receive automated recommendations, since the volume, velocity and variety of such big data is directly proportional to the amount of time users would spend browsing through it manually.

1.2 Recommender System

In the current progressive situation of information and technology, recommender systems become an interesting researching aspect and have been studied in a lot of application fields, from online shopping platforms, books to music and videos, etc. For such wide range of application fields, the term "recommender system" needs to be defined clearly from the beginning. Similar to the rocketing evolution of recommender system, its definition has also changed over the past decades.

The concept of recommender system was once defined as a system whose inputs are recommendations from users, from which it then accumulates the information and delivers it to matching recipients who are also users of the system [21]. According to Resnick and Varian, in their concept, the value of a recommender system came from its capability of generating useful matches between recommenders and people who looked for recommendations of items or services using the system. In this definition, the main objective of a recommender system was to strengthen the nexus among users within the system.

Later on, in one article of Burke (2002) [7], a recommender system is defined as any system that is able to give out outputs as personalized recommendations or capable of providing users with personalized navigation to objects, items or services that are in their interest among other various available options. By that definition,

a recommender system is expected to receive random and raw data from users' searches and purchases as inputs, process it into useful information and then deliver outputs as helpful recommendations based on users' preferences and interests.

Recommender systems thrive largely on machine learning algorithms that are employed to provide relevant suggestions to users based on their own actions and those of other users interested in similar products. Recommenders are very popular on e-commerce websites where receiving an automated recommendation makes browsing the content much easier for an end user. Rather than always having to search through an enormous range of items, it is useful if the system is able to suggest other items similar to the one that a given user just bought or browsed, based on those that are frequently bought with the given item. This is extremely helpful as new items are added constantly. Not having recommendations would make it harder for users to find all they need. While this is helpful to individual users, it is noticed that companies make a substantial percentage of their revenue through user recommendations. For example, Amazon.com states that 40% of their sales increased using their recommendation engine [30].

Amazon uses a recommender system to match each of a user's purchased and rated items to similar items which then appear to the user as recommendation lists. Some types of recommendation lists on *Amazon.com* are: a "frequently bought together" suggestion, as shown in Fig.1.1, based on pertinent items that other users generally buy along with the items in the given user's shopping cart and browsed items; a "recommended for you" list based on a previous purchase by the same user, see Fig.1.2; and a "related to items you have viewed" list based on items similar to those the given user has already seen (Fig.1.3). Their real time recommendations given by item-to-item collaborative filtering are surely powerful e-commerce methods.

Although this continues to be a constantly developing area, there are some

popular data mining techniques prevalent in building good recommender systems. As is widely known, data mining is the process of discovering interesting patterns and trends from large amounts of data where the data sources can include databases, data warehouses, the web, various other information repositories, and data streamed into a system dynamically [12]. Among the various data mining techniques, a widespread one is association rule mining with numerous applications in many domains. Collaborative filtering is yet another technique that is useful in capturing user and item preferences and hence is helpful in the design of recommender systems. In this paper, we conduct an exploratory study on these two techniques in order to build a simple baseline recommender system in e-commerce that can be scaled to other larger data.

Figure 1.1: Amazon's "Frequently Bought Together" Feature

1.3 Classification

As mentioned in the previous sections, since recommender systems have to cover such complicated and prolonged processes, it is hard to design them based on several purposes and emphasis on included components. Therefore, they are divided into

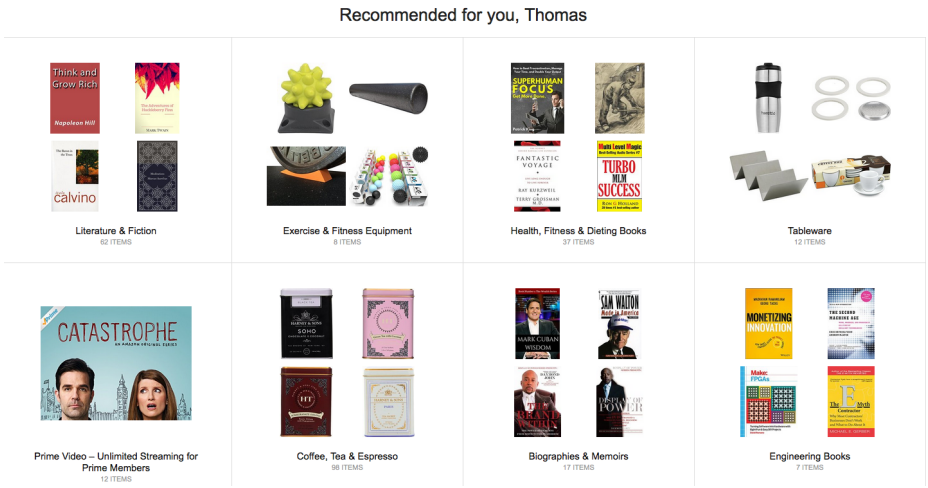

Figure 1.2: Amazon's "Recommended for you" feature

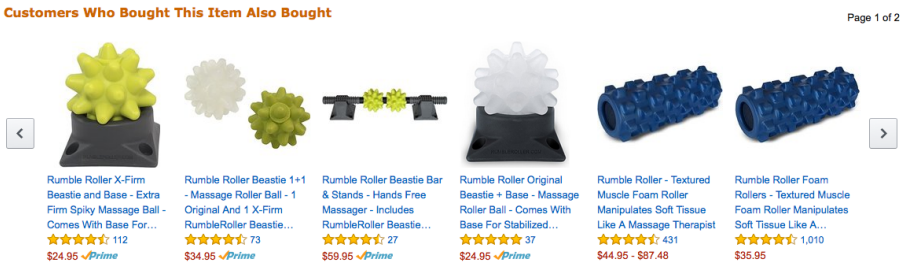

Figure 1.3: Amazon's "Related to items you have viewed" feature

three different categories, namely, memory-based recommender system, model-based recommender system and hybrid recommender system [5].

A memory-based recommender system is built to utilize the affinity metrics so as to achieve the distance between two different items or users. The second category, the model-based recommender system works on the principles of collecting, classifying and analyzing demographic factors of users or items, such as structure or characteristics, to build up a model [33]. Besides, the models are also based on the content or aggregated information provided by the users and items to get more all-round in different facets. A hybrid recommender system involves both of the two models of recommender system in order to perform better with main subset of applicable approaches, which are knowledge-based approach and collaborative

filtering. Here, we apply association rule (Chapter 3) and collaborative filtering (Chapter 4) on an Amazon shopping dataset presented in Chapter 2. In Chapter 5, we discuss sentiment classification methods and in Chapter 6 we present conclusions and future work.

CHAPTER 2

The Dataset: Amazon_US_Reviews

2.1 Overview

Amazon Customer Reviews (a.k.a. Product Reviews) is one of Amazons iconic products. In a period of over two decades since the first review in 1995, millions of Amazon customers have contributed over a hundred million reviews to express opinions and describe their experiences regarding products on the Amazon.com website. This makes Amazon Customer Reviews a rich source of information for academic researchers in the fields of Natural Language Processing (NLP), Information Retrieval (IR), and Machine Learning (ML), amongst others. Accordingly, Amazon released this data to further research in multiple disciplines related to understanding customer product experiences. Specifically, this dataset was constructed to represent a sample of customer evaluations and opinions, variation in the perception of a product across geographical regions, and promotional intent or bias in reviews.

Over 130+ million customer reviews are available to researchers as part of this release. The data is available in TSV files in the amazon-reviews-pds S3 bucket in AWS US East Region. Each line in the data files corresponds to an individual review (tab delimited, with no quote and escape characters).

The dataset we are using is the Amazon Home Improvements, Kitchen, and

Personal Care product reviews which is a subset of the large Amazon Product review. The dataset is already stored in the TensorFlow database and can be loaded directly using the *tfds* API from Tensorflow. Once the dataset is loaded, we convert it into a *Pandas* data frame using ***tfds.as_dataframe*** API.

Each row contains the following information:

Column name	**Description**
Marketplace	Two-letter country code of the marketplace where the review was written.
Customer_id	Random identifier that can be used to aggregate reviews written by a single author.
Review_id	The unique ID of the review.
Product_id	The unique Product ID the review pertains to. In the multilingual dataset the reviews for the same product in different countries can be grouped by the same product_id.
Product_parent	Random identifier that can be used to aggregate reviews for the same product.
Product_title	Title of the product.
Product_category	Broad product category that can be used to group reviews (also used to group the dataset into coherent parts).
Star_rating	The 1-5 star rating of the review.
Helpful_votes	Number of helpful votes.
Total_votes	Number of total votes the review received.
Vine	Review was written as part of the Vine program.
Verified_purchase	Indicates whether the review is on a verified purchase.
Review_headline	The title of the review.
Review_body	The review text.
Review_date	The date the review was written.

Table 2.1: Dataset's column description

2.2 Data Description

Before implementing the techniques on the dataset, it is important to explore the data in order to comprehend the attributes and fathom the content of the data. Here

we use Python on Jupiter Notebook. Specifically, it is useful to get information regarding the number of customers, reviews per customer, number of products, reviews per product and information about the ratings. In order to attain this, we first check for the unique number of customers and products. Fig. 2.1 depicts the retrieval of those numbers through a function which counts the unique values from a list. We also calculate the average number of reviews per customer and per product.

```
print('Number of unique customers: ', df.customer_id.nunique())
print('Number of unique products: ', df.product_id.nunique())
print('Review per customer: ', len(df)/df.customer_id.nunique())
print('Review per product: ', len(df)/df.product_id.nunique())
```

```
Number of unique customers:  1285915
Number of unique products:  450860
Review per customer:  1.5069798548115545
Review per product:  4.2981147141019385
```

Figure 2.1: Sample calculations for number of unique customers and products as well as average number of reviews per customer and per product

In this project, we focus on *Home Improvements*, *Kitchen*, and *Personal Care* categories as these are two of the best-selling categories on Amazon [9]. To ensure the quality of review's ratings, only reviews that are verified purchase are considered. In the data shown herewith, the number of unique customers is 1,285,915, the number of unique products is 450,860, the average number of reviews per customer is 1.51 and the average number of reviews per product is 4.30. While this is a small sample, it is well-representative of the data and presents an example of how a subset can be useful in dealing with much larger datasets. This will be helpful later in our further work on extending this study to larger datasets and hence is important from a scalability angle.

Thereafter, the summary statistics is retrieved pertaining to the ratings column in the data-frame. As observed in Fig. 2.2, the total count of ratings is 1,937,848 with a mean of 395, standard deviation of 1.49, minimum rating of 1 perceived as

poor quality, and maximum rating of 5 indicating good quality. Also, more than half of the products have a rating of 5 which is good. As depicted in Fig. 2.3, a histogram is created to view distribution of the ratings for at-a-glance analysis.

```
df['star_rating'].describe()
```

```
count    1.937848e+06
mean     3.954521e+00
std      1.487974e+00
min      1.000000e+00
25%      3.000000e+00
50%      5.000000e+00
75%      5.000000e+00
max      5.000000e+00
Name: star_rating, dtype: float64
```

Figure 2.2: Summary of statistics on the *star_rating* column

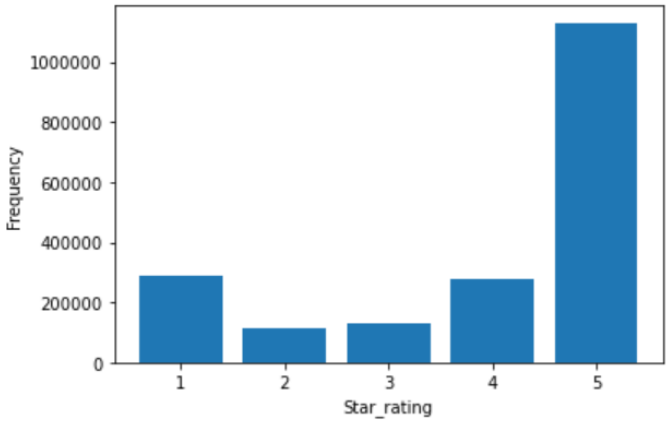

Figure 2.3: Distribution of ratings histogram

CHAPTER 3

Association Rules

In this chapter we present the theory behind association rules and then we apply it to our Amazon dataset described in previous chapter.

3.1 Introduction

Association Rule is one of the most common data mining techniques used to model market basket analysis. Retailers use this to increase sales by better understanding customer purchasing patterns. In daily life, supermarket for example, dairy items are placed in the same aisle, fresh fruits and vegetables are in the same shopping area, and beverages form another set. Organizing items in a scientific manner not only saves the customers' shopping time but also increases stores' profit as it encourages customer cross-item buying. Association rules help reveal such relationships between items. However, a drawback of such technique is that it does not take transactions at different times into account. All items corresponding to a unique customer bought in one transaction event are grouped as one set. Market basket analysis is helpful to recommend the products that are likely to be purchased together based on a set of products that are currently in the basket.

Association Rules technique consists of an antecedent and a consequent, both of which are lists of items [32]. Note that the relationship implied in this technique

is co-occurrence, not causality. ***Itemset*** is the list of items included in both the antecedent and the consequent. For example, the itemset comprising of bread, egg, and milk can generate the antecedents and consequents illustrated in Table 3.1, denoted as $\{X\} \rightarrow \{Y\}$. For instance, the first entry in this table models the event that customer who bought bread and egg also bought milk, i.e. $\{Bread, Egg\} \rightarrow \{Milk\}$. Note that the antecedents and consequents are not symmetric and we will see why that is the case in section *Confidence*.

Antecedent	**Consequent**
Bread, Egg	Milk
Bread, Milk	Egg
Bread	Egg, Milk
Egg, Milk	Bread
Egg	Bread, Milk
Milk	Bread, Egg

Table 3.1: Possible antecedents and consequents from itemset {Bread, Egg, Milk}

To demonstrate the use of the support-confidence framework, an example of the process of mining association rules is outlined below. Table 3.2 illustrates five customers shopping in a grocery store and the items that each of them bought. In this project, we group all items that a customer bought together and perform analysis on it. Thus, we consider that each unique customer performs only one transaction as we do not have information about different purchases done on different date.

Customer ID	**Itemset**
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Table 3.2: Example of an itemset (or transaction) database

3.2 Support

The support of Association Rule denoted by $Support(\{X\})$ is a measure of how frequently an *itemset* occurs among all transactions [3]. A higher value (i.e., closer to one) of support indicates that the itemset is more popular to consumers.

Definition 3.1 Support gives an idea of how frequent an itemset is in all the transactions.:

$$\text{Support}(\{X\}) = \frac{\text{Number of customers who bought X}}{\text{Number of unique customers}} \tag{3.1}$$

To illustrate equation (3.1), let consider the itemset $\{Bread, Milk\}$, which occurs three times in the customers' baskets (in transactions $1, 4, 5$). With five customers in total, the computation of the support for $\{Bread, Milk\}$ is as follow.

$$\text{Support}(\{Bread, Milk\}) = \frac{\text{Number of customers who bought \{Bread, Milk\}}}{\text{Number of unique customers}} = \frac{3}{5} = 0.6 \tag{3.2}$$

An itemset in the database is called frequent if its support is equal or greater than the threshold minimum support given by users. In this particular example, to assure that an itemset occurs at least twice, it is reasonable to set the minimum threshold of 0.4. Thus, {Bread, Milk} is a frequent itemset. Consequently, the threshold is expected to get smaller as the dataset gets larger. Since *Support* is another term for frequency, its value ranges from zero (indicating the item does not appear in the dataset) to one (implying the item is bought in every transaction).

3.3 Confidence

This measure indicates the strength of correlation between X and Y in the database. This is to define how often customers, for instance, buy $\{Coke\}$ when there is

already $\{Bread, Milk\}$ in their cart. Mathematically, confidence is the conditional probability of a consequent's appearance given an antecedent. In addition, although $(\{Bread, Milk\} \rightarrow \{Coke\})$ and $(\{Bread\} \rightarrow \{Milk, Coke\})$ indicate the same itemset of $\{Bread, Milk, Coke\}$, the confidence level is affected by the change of the antecedent.

Definition 3.2 Confidence defines the likeliness of occurrence of consequent on the cart given that the cart already has the antecedents.:

$$\text{Confidence}(\{X\} \rightarrow \{Y\}) = \frac{\text{Number of customers who bought both X and Y}}{\text{Number of customers who bought X}} \tag{3.3}$$

For instance, there are two customers buying $\{Coke\}$ out of three who already had $\{Bread, Milk\}$ in their basket.

$$\begin{aligned} &\text{Confidence}(\{Bread, Milk\} \rightarrow \{Coke\}) \\ &= \frac{\text{Number of customers who bought both \{Bread, Milk\} and \{Coke\}}}{\text{Number of customers who bought \{Bread, Milk\}}} \\ &= \frac{2}{3} = 0.67 \end{aligned} \tag{3.4}$$

Nevertheless, confidence value for $\{Coke\}$ will always be high with most of the antecedents due to its presence in four out of five carts listed above in Table 3.2. Let introduce an example where this might be a drawback.

As shown in Fig.3.1, suppose a store sold a total of 80 bottles of milk and 16 bottles of shampoo, ten of which are bought together. Note that in this example, each customer bought only one unit of each product at most.

$$\begin{aligned} &\text{Confidence}(\{Shampoo\} \rightarrow \{Milk\}) \\ &= \frac{\text{Number of customers bought both \{Shampoo\} and \{Milk\}}}{\text{Number of customers bought \{Shampoo\}}} \\ &= \frac{10}{10 + 6} = 0.625 \end{aligned} \tag{3.5}$$

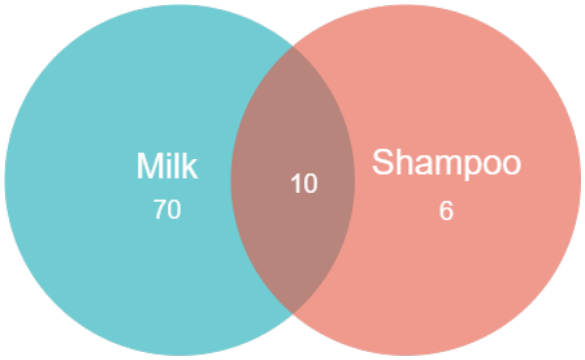

Figure 3.1: Quantity of milk and shampoo sold in a convenient store

Confidence value ranges from zero to one since it is the probability of a customer buying $\{X\}$ after buying $\{Y\}$. The result from Equation 3.5 is a high confidence value, but intuitively these two products have a weak correlation, or association. Here, milk is a product bought very frequently, unlike shampoo, and the confidence measure does not capture this. It is evident that high confidence value can sometimes be misleading, so another measure, *Lift*, is introduced to solve this issue.

3.4 Lift

Lift overcomes the misleading frequency problem illustrated above by controlling the *Support* of consequent while computing the *Confidence* of itemset $\{Y\}$ given itemset $\{X\}$.

Definition 3.3 Lift controls for the *support* (frequency) of consequent while calculating the conditional probability of occurrence of itemset Y given itemset X:

$$Lift(\{X\} \rightarrow \{Y\}) = \frac{Confidence(\{X\} \rightarrow \{Y\})}{Support(\{Y\})} \tag{3.6}$$

It is implied from Equation 3.6 that the greater the value of *Lift*, the greater the association between $\{X\}$ and $\{Y\}$. On the other hand, if *Lift* is less than 1, we expect

that $\{Y\}$ is less likely to be in cart given the presence of $\{X\}$. The following example provides a better understanding of how *Lift* works.

The Confidence($\{Shampoo\} \rightarrow \{Milk\}$) is 0.625 as calculated above. Suppose Support($\{Milk\}$, or the probability of having milk on the cart without any knowledge about shampoo, is 0.8.

$$\text{Lift}(\{Shampoo\} \rightarrow \{Milk\}) = \frac{\text{Confidence}(\{Shampoo\} \rightarrow \{Milk\})}{\text{Support}(\{Milk\})} = \frac{0.625}{0.8} = 0.78 \quad (3.7)$$

Consequently, the *Lift* for milk given shampoo is 0.78, which provides a more realistic picture. A value of *Lift* less than 1 indicates that having shampoo in the cart does not increase the probability for buying milk. This helps preventing from *Confidence*'s bias conclusion. *Lift* value is non-negative as none of its component can be less than zero, but it has no upper bound limit because the value increases as $Confidence(\{X\} \rightarrow \{Y\})$ becomes larger or $Support(\{y\})$ decreases. The measure of *Lift* supports store managers to decide product organizing on aisle.

3.5 Association Rule Mining

3.5.1 One-Hot Encoding

Apriori algorithm requires transforming the original database to a dataframe which includes only 0 and 1 or True and False as data to begin generating possible candidate itemsets. The technique used is called One-Hot Encoding [8], which has all unique items as columns and each row as a unique customer. Observations with value of 1 represent an item that was bought by the customer while 0 denotes that the item was not bought. The following table is an One-Hot Encoded Boolean table created from the itemset database of Table 3.2.

Customer ID	Bread	Milk	Diaper	Beer	Coke	Eggs
1	1	1	0	0	0	0
2	1	0	1	1	0	1
3	0	1	1	1	1	0
4	1	1	1	1	0	0
5	1	1	1	0	1	0

Table 3.3: One-Hot Encoding data from Table 3.2

3.5.2 Generating Itemsets from a List of Items

The next step in building an Association Rule model, is generating the *frequent* itemsets from the transactions. For simplicity, suppose a store only sells *Bread, Milk, Coke, Beer*. Then, the size of the possible itemsets vary from one to the number of unique items, which is four. All possible itemsets of size 1, 2, 3, 4 are enumerated in Fig.3.2.

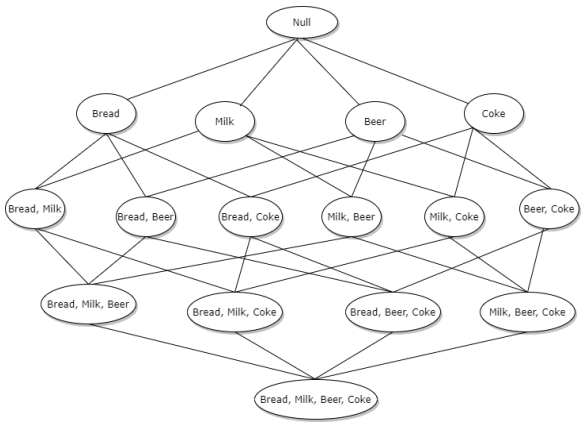

Figure 3.2: Possible itemsets generated from {*Bread*, *Milk*, *Coke*, *Beer*}

The next step is to compute the *Support* using Equation 3.1 of each itemset. Those with higher *Support* value than the minimum threshold are called *frequent* itemsets.

3.5.3 Generating all Possible Association Rules from the Frequent Itemsets

A brute force called ***Apriori principle*** is a common approach to efficiently help forming all possible itemsets and check the *Support* value for each of them. Apriori principle prunes all the *supersets* of an itemset that does not satisfy the minimum threshold. For example, if the Support value of $\{Beer, Coke\}$ is lower than the set threshold, then also any itemset containing beer and coke does not cross the threshold. Furthermore, this technique generates new itemsets by adding another item to the current frequent itemsets and checking on the Support values repetitively. The Apriori principle is shown in Fig.3.3 below.

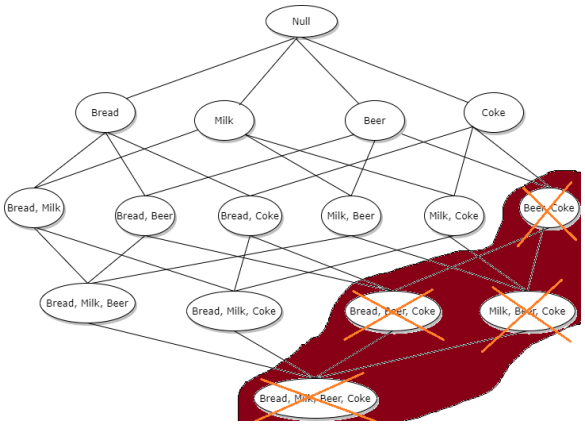

Figure 3.3: Applying Apriori algorithm to the imtesets generated from $\{Bread, Milk, Coke, Beer\}$. If itemset $\{Beer, Coke\}$ is not frequent, then supersets are also infrequent, and thus, can be eliminated from future conclusions.

From the candidate rules, it is essential to identify rules which satisfy the confidence level threshold. Applying Apriori principle to confidence of rules as with the itemsets' support value, it is efficient to neglect the supersets of a consequent once its confidence falls below the threshold. In other words, confidence of $\{A, B, C \rightarrow D\} \geq \{B, C \rightarrow A, D\} \geq \{C \rightarrow A, B, D\}$. As Confidence($\{X\} \rightarrow \{Y\}$) = Support$\{X, Y\}$/Support$\{X\}$, the numerator remains constant for the same itemset

while the support of X increases as its size decreases, which in turn cause the confidence to decrease.

3.5.4 Association Rule Pseudo Code

The pseudo code for the frequent itemset generation part of the *Apriori* algorithm is shown in Fig. 3.4 [27]. Let C_k denote the set of candidate 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 1-frequent itemsets, F_1, will be known (steps 1 and 2).

- Next, the algorithm will iteratively generate new candidate k-itemsets using the frequent $(k-1)$-itemsets found in the previous iteration (step 5). Candidate generation is implemented using a function called apriori-gen, whose algorithm is described in Section 3.5.3.

- To count the support of the candidates, the algorithm needs to make an additional pass over the date set (steps 6-10). The subset function is used to determine all the candidate itemsets in C_k that are contained in each transaction t. The implementation of this function is based on the description in Section 3.2.

- After counting their supports, the algorithm eliminates all candidate itemsets whose suppport counts are less than **minsupp** (step 12).

- The algorithm terminates when there are no new frequent itemsets generated (step 13).

1: $k = 1$.
2: $F_k = \{ i \mid i \in I \wedge \sigma(\{i\}) \geq N \times minsup\}$. {Find all frequent 1-itemsets}
3: **repeat**
4: $k = k + 1$.
5: $C_k = \text{apriori-gen}(F_{k-1})$. {Generate candidate itemsets}
6: **for** each transaction $t \in T$ **do**
7: $C_t = \text{subset}(C_k, t)$. {Identify all candidates that belong to t}
8: **for** each candidate itemset $c \in C_t$ **do**
9: $\sigma(c) = \sigma(c) + 1$. {Increment support count}
10: **end for**
11: **end for**
12: $F_k = \{ c \mid c \in C_k \wedge \sigma(c) \geq N \times minsup\}$. {Extract the frequent k-itemsets}
13: **until** $F_k = \emptyset$
14: Result $= \bigcup F_k$.

Figure 3.4: Frequent itemset generation of the *Apriori* algorithm [27]

3.5.5 The Pros and Cons of Association Rule

Apriori is the basic and most popular algorithm proposed by R. Agrawal and R. Srikant for finding frequent itemsets based on candidate generation [3]. Candidates are itemsets containing all frequent itemsets. The name of the algorithm Apriori is based on the Apriori property which states that all nonempty subsets of a frequent itemset must also be frequent. The core step of the algorithm is generation of candidate k-itemsets C_k from frequent $(k-1)$-itemsets, L_{k-1}, and it consists of *join* and *prune* actions. In join step, the conditional join of $L_{k-1} \times L_{k-1}$ is assigned to candidate set C_k while prune step reduces the size of C_k using Apriori property [12].

One of the most important mechanisms in the Apriori algorithm is the use of the hash tree data structure. It uses this data structure in the candidate support counting phase to reduce the time complexity from $O(kmn)$ to $O(kmT + n)$, where k is the average size of the candidate itemset, m represents the number of candidates, n represents the number of items in the whole dataset, and T is the number of transactions.

The major advantage of the Apriori algorithm comes from its memory usage because only the $k-1$ frequent itemsets, L_{k-1}, and the candidates in level k, C_k, need to be stored in the memory. It generates the minimum number of candidates based

on the $L_{k-1} \times L_{k-1}$ (described in [3]) and the pruning method, and it stores them in the compact hash tree structure. In case the candidates fill up the memory from the dataset and a low minsup setting, the Apriori algorithm does not generate all the candidates to overload the memory. Instead, it generates as many candidates as the memory can hold. The Apriori algorithm, unfortunately, have to go through the expensive candidate generation and support counting process. This causes a disadvantage in running time [16].

When the size of the database is very large, the Apriori algorithm will fail. because large database will not fit with memory (RAM). Thus, each pass requires large number of disk reads. For example, 1 GB database stored in hard disk with block size 8KB require roughly 125,000 block reads for a single pass. For 10 passes require roughly 1,250,000 block reads. Assume it takes 12 ms to read a single block. That means it takes roughly 3.5 hours time for entire block reads. Because of the time and memory limitation of a personal computer, this project will first apply the Association Rule so that it derive rules of the form $\{A\} \rightarrow \{B\}$, where A and B contain only one item each.

3.6 Association Rules Mining Using Python Generators to Handle Large Datasets

Since learning the relationships between any given pair of items is the main focus of this part, using apriori to get to item sets of size two is sufficient. Going through various iterations, we split the data into multiple subsets to get functions like crosstab and combinations to run on a machine with 8 GB of RAM memory. But even with this approach, the computer could only process about 1800 items before the kernel would crash. To solve this problem, we shall use Python generators.

3.6.1 Python Generators

In a nutshell, a generator is a special type of function that returns an iterable sequence of items. However, unlike regular functions which return all the values at once (e.g. returning all the elements of a list), a generator *yields* one value at a time. To get the next value in the set, we must ask for it - either by explicitly calling the generator's built-in "next" method, or implicitly via a for loop. This is a great property of generators because it means that we don't have to store all of the values in memory at once. We can load and process one value at a time, discard when finished and move on to process the next value. This feature makes generators perfect for creating item pairs and counting their frequency of co-occurence. Here is a concrete example of what we are trying to accomplish:

1. Get all possible item pairs for a given order

   Customer 1: bread, egg, milk → item pairs: {bread, egg}, {bread, milk}, {egg, milk}

   Customer 2: egg, milk → item pairs: {egg, milk}

2. Count the number of times each item pair appears

   {bread, egg}: 1

   {bread, milk}: 1

   {egg, milk}: 2

3. Generate candidate itemsets and compute support values to find frequent itemsets (support ≥ *minsupp*). Measure and evaluate confidence and lift values to generate association rules.

3.7 Scalable One-to-One Association Rules Mining Demonstration

Before applying Association Rules to such a large dataset, it is essential to test the program with a smaller controlled dataset. Therefore, we created an "artificial" sample by taking the first twelve rows from the original dataset and replicate it for four times. To make our sample more realistic with a variety of customers buying the same range of goods, we add a different letter to the customers' ID each time the original rows are duplicated. Eventually, our make-up dataset consists of 40 customers and 11 distinguished items which is shown in Fig.3.5.

Customer ID	Item ID	Customer ID	Item ID	Customer ID	Item ID	Customer ID	Item ID	Customer ID	Item ID
Customer 1	Item 1	Customer 1a	Item 1	Customer 1b	Item 1	Customer 1c	Item 1	Customer 1d	Item 1
Customer 2	Item 2	Customer 2a	Item 2	Customer 2b	Item 2	Customer 2c	Item 2	Customer 2d	Item 2
Customer 2	Item 3	Customer 2a	Item 3	Customer 2b	Item 3	Customer 2c	Item 3	Customer 2d	Item 3
Customer 2	Item 6	Customer 2a	Item 6	Customer 2b	Item 6	Customer 2c	Item 6	Customer 2d	Item 6
Customer 3	Item 5	Customer 3a	Item 5	Customer 3b	Item 5	Customer 3c	Item 5	Customer 3d	Item 5
Customer 4	Item 6	Customer 4a	Item 6	Customer 4b	Item 6	Customer 4c	Item 6	Customer 4d	Item 6
Customer 4	Item 7	Customer 4a	Item 7	Customer 4b	Item 7	Customer 4c	Item 7	Customer 4d	Item 7
Customer 5	Item 8	Customer 5a	Item 8	Customer 5b	Item 8	Customer 5c	Item 8	Customer 5d	Item 8
Customer 6	Item 9	Customer 6a	Item 9	Customer 6b	Item 9	Customer 6c	Item 9	Customer 6d	Item 9
Customer 7	Item 10	Customer 7a	Item 10	Customer 7b	Item 10	Customer 7c	Item 10	Customer 7d	Item 10
Customer 8	Item 11	Customer 8a	Item 11	Customer 8b	Item 11	Customer 8c	Item 11	Customer 8d	Item 11
Customer 8	Item 12	Customer 8a	Item 12	Customer 8b	Item 12	Customer 8c	Item 12	Customer 8d	Item 12

Figure 3.5: Testing artificial data for Association Rules program

The next step of pre-processing data is to group the items bought by customer, as in Fig. 3.6.

3.7.1 itertools.combinations(iterable, r)

This Python method returns *r* length subsequences of elements from the input *iterable* [1].

The combination tuples are emitted in lexicographic ordering according to the order of the input *iterable*. In other words, if the input *iterable* is sorted, the combination tuples will be produced in sorted order.

Customer ID	Item ID
Customer 1	Item 1
Customer 2	Item 2, Item 3, Item 6
Customer 3	Item 5
Customer 4	Item 6, Item 7
Customer 5	Item 8
Customer 6	Item 9
Customer 7	Item 10
Customer 8	Item 11, Item 12

Customer ID	Item ID
Customer 1a	Item 1
Customer 2a	Item 2, Item 3, Item 6
Customer 3a	Item 5
Customer 4a	Item 6, Item 7
Customer 5a	Item 8
Customer 6a	Item 9
Customer 7a	Item 10
Customer 8a	Item 11, Item 12

Customer ID	Item ID
Customer 1b	Item 1
Customer 2b	Item 2, Item 3, Item 6
Customer 3b	Item 5
Customer 4b	Item 6, Item 7
Customer 5b	Item 8
Customer 6b	Item 9
Customer 7b	Item 10
Customer 8b	Item 11, Item 12

Customer ID	Item ID
Customer 1c	Item 1
Customer 2c	Item 2, Item 3, Item 6
Customer 3c	Item 5
Customer 4c	Item 6, Item 7
Customer 5c	Item 8
Customer 6c	Item 9
Customer 7c	Item 10
Customer 8c	Item 11, Item 12

Customer ID	Item ID
Customer 1d	Item 1
Customer 2d	Item 2, Item 3, Item 6
Customer 3d	Item 5
Customer 4d	Item 6, Item 7
Customer 5d	Item 8
Customer 6d	Item 9
Customer 7d	Item 10
Customer 8d	Item 11, Item 12

Figure 3.6: Transformed data frame from Fig. 3.5

Elements are treated as unique based on their position, not on their value. So if the input elements are unique, there will be no repeat values in each combination.

The resulted data frame (Fig.3.6) is the standard form to apply Association Rules since we can utilize the *combination* method from *itertools* library and easily generate the candidate itemsets with the length of our choice. Each row from Fig. 3.6 is an *iterable*. As mentioned above, we are more interested in only the one-to-one relationship between items, which has the length of two ($r = 2$). Below is the demonstration of how *combination* method executes.

Iterable := {Item 2, Item 3, Item 6}

→Combinations(Iterable, 2) : {Item 2, Item 3}, {Item 2, Item 6}, {Item 3, Item 6}

→Combinations(Iterable, 3) : {Item 2, Item 3, Item 6}

(3.8)

3.7.2 Demonstration Result

Initially, we measure the *support* value of each item and compare it to our set *min-supp*. Here, we set threshold to 0.0005 because we want to double-check all

possible combinations of item pairs. As a result, all eleven different items pass the *min-supp* of 0.0005. Then, we filter out for customers who bought more than two items and at least one of which yields the *support* value higher than the set *min-supp*. There are fifteen customers buying more than two products in our sample and five item pairs are formed. Summary of the result is shown in Fig. 3.7. Equation 3.10 lists the customers and item pairs generated from the sample's Association Rules. This artificial data set illustrates that our Association Rules implementation identifies correctly itemsets of size two, having *support* larger than the threshold 0.0005.

Customers who bought two or more items:

Customer 2, Customer 4, Customer 8, Customer 2a, Customer 4a, Customer 8a
Customer 2b, Customer 4b, Customer 8b, Customer 2c, Customer 4c, Customer 8c
Customer 2d, Customer 4d, Customer 8d

(3.9)

Item pairs:

{Item 2, Item 3}, {Item 2, Item 6}, {Item 3, Item 6}, {Item 6, Item 7}, {Item 11, Item 12}

(3.10)

```
%%time
rules = association_rules(orders_test, 0.0005)
```

```
Starting orders:                     60
Items with support >= 0.0005:        11
Remaining orders:                    60
Remaining customers with 2+ items:   15
Remaining orders:                    35
Item pairs:                           5
Item pairs with support >= 0.0005:    5

Wall time: 17.9 ms
```

Figure 3.7: Number of orders, items, and item pairs that meet the requirements

3.7.3 Results

After validating our implementation of Association Rule mining on the sample data, we then apply it to the main data. Eventually, we only show rows with *Lift* greater than one in our results since they indicate that the antecedent and consequent are highly correlated and are more likely to be bought together (as mentioned in Section 3.4).

By experimenting with the Association Rule on the product purchase history for the category of Health & Personal Care and Home & Kitchen from Amazon review data, we discover some association rules. These rules can be used to provide suitable recommendations through the users' purchase history by recommending what they might want to buy next or by recommending additional products to the ones they are considering buying. The recommendations through association rules are all based on patterns that previous users have created through buying products. These recommendations can be extremely helpful to a given user and are likely to bring more profit to companies implementing them in recommendation systems.

item_A	item_B	freqAB	supportAB	freqA	supportA	freqB	supportB	confidenceAtoB	confidenceBtoA	lift
b'B00I9J7KSY'	b'B00HK6B9X4'	206	0.006928	715	0.024047	602	0.020247	0.288112	0.342193	14.229951
b'B00N277J2A'	b'B00N26M2F0'	205	0.006895	547	0.018397	740	0.024888	0.374771	0.277027	15.058217
b'B00N26M2F0'	b'B00N277J2A'	201	0.006760	740	0.024888	547	0.018397	0.271622	0.367459	14.764398
b'B00I9J7KSY'	b'B00J7HQ1L4'	192	0.006457	715	0.024047	843	0.028352	0.268531	0.227758	9.471229
b'B00J7HQ1L4'	b'B00HK6B9X4'	188	0.006323	843	0.028352	602	0.020247	0.223013	0.312292	11.014696
b'B00HK5RFBK'	b'B00HK6B9X4'	186	0.006256	485	0.016312	602	0.020247	0.383505	0.308970	18.941460
b'B00HK6B9X4'	b'B00I9J7KSY'	181	0.006088	602	0.020247	715	0.024047	0.300664	0.253147	12.503016
b'B004KQ9HZY'	b'B009YZN8G0'	167	0.005617	373	0.012545	465	0.015639	0.447721	0.359140	28.628159
b'B009YZN8G0'	b'B004KQ9HZY'	164	0.005516	465	0.015639	373	0.012545	0.352688	0.439678	28.113880
b'B00J7HQ1L4'	b'B00HK5RFBK'	162	0.005448	843	0.028352	485	0.016312	0.192171	0.334021	11.781062

Figure 3.8: Initial outputs from association rule mining

Fig. 3.8 shows a few sample outputs from association rules. It includes two products and the calculated support, confidence and lift values. The products are shown in the first two columns with their product IDs. The product IDs

itemA	itemB	freqAB	supportAB	freqA	supportA	freqB	supportB	confidenceAtoB	confidenceBtoA	lift
b'Ubervita Roct Pro Extreme Workout Ignitor Th...	b'Ubervita Ubersurge Endurance Pre Workout Ene...	206	0.006928	715	0.024047	602	0.020247	0.288112	0.342193	14.229951
b'Nexgen Biolabs Xentrafen PM Thermogenic Non-...	b'Nexgen Biolabs Xentrafen Maximum Strength Di...	205	0.006895	547	0.018397	740	0.024888	0.374771	0.277027	15.058217
b'Nexgen Biolabs Xentrafen Maximum Strength Di...	b'Nexgen Biolabs Xentrafen PM Thermogenic Non-...	201	0.006760	740	0.024888	547	0.018397	0.271622	0.367459	14.764398
b'Ubervita Roct Pro Extreme Workout Ignitor Th...	b'Ubervita Ubertest All Natural Testosterone B...	192	0.006457	715	0.024047	843	0.028352	0.268531	0.227758	9.471229
b'Ubervita Ubertest All Natural Testosterone B...	b'Ubervita Ubersurge Endurance Pre Workout Ene...	188	0.006323	843	0.028352	602	0.020247	0.223013	0.312292	11.014696
b"Ubervita Uberday Men's Multivitamin, Superio...	b'Ubervita Ubersurge Endurance Pre Workout Ene...	186	0.006256	485	0.016312	602	0.020247	0.383505	0.308970	18.941460
b'Ubervita Ubersurge Endurance Pre Workout Ene...	b'Ubervita Roct Pro Extreme Workout Ignitor Th...	181	0.006088	602	0.020247	715	0.024047	0.300664	0.253147	12.503016
b"Hydroxycut Pro Clinical, America's Number 1 ...	b'Muscletech Hydroxycut Nutrition Gummies, Mix...	167	0.005617	373	0.012545	465	0.015639	0.447721	0.359140	28.628159
b'Muscletech Hydroxycut Nutrition Gummies, Mix...	b"Hydroxycut Pro Clinical, America's Number 1 ...	164	0.005516	465	0.015639	373	0.012545	0.352688	0.439678	28.113880
b'Ubervita Ubertest All Natural Testosterone B...	b"Ubervita Uberday Men's Multivitamin, Superio...	162	0.005448	843	0.028352	485	0.016312	0.192171	0.334021	11.781062

Figure 3.9: Mapping of product IDs to product names

are then mapped in our program to the corresponding product names using the literature on these products. A snapshot of this mapping is presented in Fig. 3.9 with the respective products. These values can therefore be used to stipulate recommendations via the users' purchase history by suggesting what they should buy next or by suggesting other relevant products when they are considering a certain product. Such recommendations are incorporated into the output panel of our simple baseline e-commerce recommender system to offer meaningful suggestions to users about products. Using the first row in Fig. 3.9 as an example, the output of the association rules can be interpreted as follows:

- If a user buys product *Ubervita Roct Pro Extreme Workout Ignitor* then, based on *confidenceAtoB*, there is a 0.288 probability that user also buys *Ubervita Ubersurge Endurance Pre Workout*.

- The probability of product *Ubervita Roct Pro Extreme Workout Ignitor* occurring in the data set is 0.0069 (based on *supportAB*).

- The result from this example suggests firms to differentiate their products and

recommend to customers goods with similar usage which can complement each other.

Definition 3.4 A **complementary good** is a good whose use is related to the use of an associated or paired good. Two goods (A and B) are complementary if using more of good A requires more use of good B. [31]:

For example, the demand for one good (pre-workout supplement) generates demand for the other (post-workout supplement).

Consider the market for a newly introduced experience good such as natural heart supplement. Firms entering the market faces a problem because consumers are initially uninformed about heart supplement products. Consumers, therefore, risk being disappointed if they purchase Internet service for the first time. In theory, if all consumers were fully informed about the products, they would be willing to pay a higher price. Before their introduction, however, all consumers are uninformed about heart supplement and risk buying and disliking it. Uninformed consumers will be willing to pay less for the products than informed consumers [31]. Satisfied informed customers are willing to pay a higher price and re-buy the products of the same brand with confidence. This is the fundamental for brand loyalty.

Definition 3.5 Brand loyalty is the positive association consumers attach to a particular product or brand. Customers who exhibit brand loyalty are devoted to a product or service, which is demonstrated by their repeat purchases despite competitors' efforts to lure them away. [31]:

The brand royalty theory is strengthened by an example in the second and third rows of Fig. 3.10, where customers who bought "Ozeri Green Earth Textured Ceramic Nonstick Pan" are also confident in buying "Ozeri ZB19-w Rev Digital Bathroom Scale with Electro-Mechanical Weight Dial" and vice versa. Although these two products are from different categories (Kitchen and Health & Personal

Care), they are daily used lifestyle products for the modern home. Therefore, once they are satisfied with their first purchase with Ozeri brand, many customers believe that other kitchen, bath, entertainment, and personal amenities goods provide similar high quality usage experience.

Rows 1, 4, and 5 in Fig. 3.10 shows that customers who bought vitamin/supplement goods also have the incentive to buy stainless steel spoons to use with (complementary goods, Definition 3.4). Capturing the brand loyalty along with the example above, firms selling those personal care products should include a spoon in their future product and sell them as a bundle at a higher but reasonable price. Informed customers giving good ratings for the previous goods will still buy the bundle since they think that the spoon included can help them consume the servings with more exact amount.

itemA	itemB	freqAB	supportAB	freqA	supportA	freqB	supportB	confidenceAtoB	confidenceBtoA	lift	categoryA	categoryB
b'Natizo Stainless Steel Measuring Spoon Set -...	b'Liquid Vitamin D Drops - 2oz D3 100 Iu Per D...	38	0.001278	404	0.013588	299	0.010056	0.094059	0.127090	9.353406	b'Kitchen'	b'Health & Personal Care'
b'Ozeri ZB19-W Rev Digital Bathroom Scale with...	b'Ozeri Green Earth Textured Ceramic Nonstick ...	35	0.001177	123	0.004137	184	0.006188	0.284553	0.190217	45.981575	b'Health & Personal Care'	b'Kitchen'
b'Ozeri Green Earth Textured Ceramic Nonstick ...	b'Ozeri ZB19-W Rev Digital Bathroom Scale with...	31	0.001043	184	0.006188	123	0.004137	0.168478	0.252033	40.726538	b'Kitchen'	b'Health & Personal Care'
b'Natizo Stainless Steel Measuring Spoon Set -...	b'Athelas Neutraceuticals Natural Triple Stren...	31	0.001043	404	0.013588	204	0.006861	0.076733	0.151961	11.183787	b'Kitchen'	b'Health & Personal Care'
b'Natizo Stainless Steel Measuring Spoon Set -...	b'PRO-15 Advanced Strength Probiotics: 3x the ...	29	0.000975	404	0.013588	298	0.010023	0.071782	0.097315	7.162079	b'Kitchen'	b'Health & Personal Care'

Figure 3.10: Mapping of product IDs to product names (inter-category result)

3.8 Usefulness of Association Rule Mining

3.8.1 Company Level

Next, we give a few examples of how patterns discovered via association rules can be used by companies to improve their business. If a company would implement a Covid-19 based recommender system using association rules techniques, there would be many advantages. A company could use the patterns in user purchase history to create new promotions, e.g. if *Tzumi antibacterial wipes* are bought very frequently with *Germ-X hand sanitizers*, there could be a promo of buy two get one free in this context, provided there is enough supply of these items. These marketing strategies can be used by an online shopping site such as Amazon to jointly promote the sale of these items.

In the case of a physical store such as CVS Pharmacy, the association rules would help identify where the products should be placed on the shelves for an easier shopping experience and for product marketing to customers.

In an online aspect, a company such as Amazon could recommend items based on the item the customer currently has in their cart or has previously purchased, offering suggestions of Covid-19 related items frequently bought together. All these benefits could bring companies such as Amazon and CVS more revenue, in addition to helping them in better understanding customer behavior and conveying suggestions to product vendors on manufacturing specific items in high demand. Most importantly, this would have the significant broader impact of producing more items really needed by users based on their buying behavior.

3.8.2 Customer Level

Association rule results are also useful to customers, not only to companies. If a customer uses a recommender system that deploys association rules, focusing

on Covid-19 related transactions, the customer would have a seamless shopping experience. In times of the Covid-19 lockdown as well as in its recovery phase and the aftermath, customers are likely to spend less time in places such as grocery stores and pharmacies, preferring to be at home or outdoors in the open air. Hence, in the case of any physical store, e.g. a pharmacy, if frequently bought items, especially those much needed during this pandemic, are placed next to each other customers would have the pleasure of shopping with greater efficiency. Moreover, if a certain item is placed on a shelf instead of another similar / related item that is out-of-stock, then that would make things easier

For example, Lysol disinfectant sprays have been outof-stock for quite some time and customers have been using Windex cleaners instead on kitchen tabletops, light switches etc. Such a fact could be discovered by association rules on earlier data when both items were available, e.g. customers who bought *Lysol sprays* also bought *Windex cleaners*. Hence, placing Windex cleaners on the same shelf where Lysol sprays used to be (when in-stock) would provide customers a better, quicker shopping experience and would also save the store staff the time of attending to those customers' requests about finding similar items.

Likewise, when shopping online it is good to have items that are frequently bought together (or used in place of each other) to be recommended. This would make the customers' life easier; if they were also going to get one of the recommended items in addition to or instead of a given item, they would not have to spend much time searching. Overall, such arrangements would give the customers a more pleasant time while shopping and therefore they would be likely to shop at the same place again, online or in-person.

CHAPTER 4

Item Based Collaborative Filtering

This chapter introduces the collaborative filtering technique and apply it to the subset in which each customer bought more than 50 items.

4.1 Introduction

The idea of collaborative filtering was established quite soon in the history of computing and technology. In the early 1990s, collaborative filtering started to develop as a solution for the contemporary situation of users getting overloaded with disorganized and cumbersome display of information provided on the Internet [11]. Back then, this very innovative solution was intensely applied by big enterprises and businesses that now are pioneers in this field. For instance, since the beginning of the 21st century, starting from the 2000s, Amazon has been implementing collaborative filtering into their managing system as well as their shopping platforms for the customers, from which they succeeded in enhancing the customer service as well as shopping experience [29].

Later on, from the very basic and manual version of collaborative filtering technique, its gradual development resulted in the introduction of automated collaborative filtering systems into the technological researching field. The concept of automated collaborative filtering system differs from that of the manual one

as explained next. The automatic version locates relevant opinions based on inputs from searches or recommendations of customers, then aggregates them into useful information and outputs helpful personalized recommendations for items or services of interest. The main principles that collaborative filtering technique follows are based on the idea that the interest pattern of the users is maintained for future use as well, which means the items preferred by customers now will continue to be considered together with similar products in the future.

Generally speaking, collaborative filtering technique is defined to be an approach of setting up a recommender system, which assesses historical interactions and behaviors of the users and then regulates the link between the users and their items of interest.

Item-based collaborative filtering is one kind of recommendation method which looks for similar items based on the items users have already liked or positively interacted with. Simply put, Item Based Collaborative Filtering (IBCF) suggests an item based on items the user has previously consumed and consists of two parts. Item-based techniques first analyze the user-item matrix to identify relationships between different items, and then use these relationships to indirectly compute recommendations for users. Collaborative Filtering works by storing user preferences for things in a database. Leo, a new user, is matched against the database to find neighbors, or other users who have previously had Leo's taste preferences. Items that the neighbors enjoy are subsequently recommended to Leo, as he is likely to enjoy them as well.

Let's understand this with the example shown in (Fig. 4.1). Suppose customer A wants to purchase some different items and we need to recommend him an item based on his past preferences. We will first search for items that customer A has bought or liked, let's call those items 'A', 'B' and 'C'. Next, we will search for other items similar to those three products. Suppose we found out that item 'D' is highly

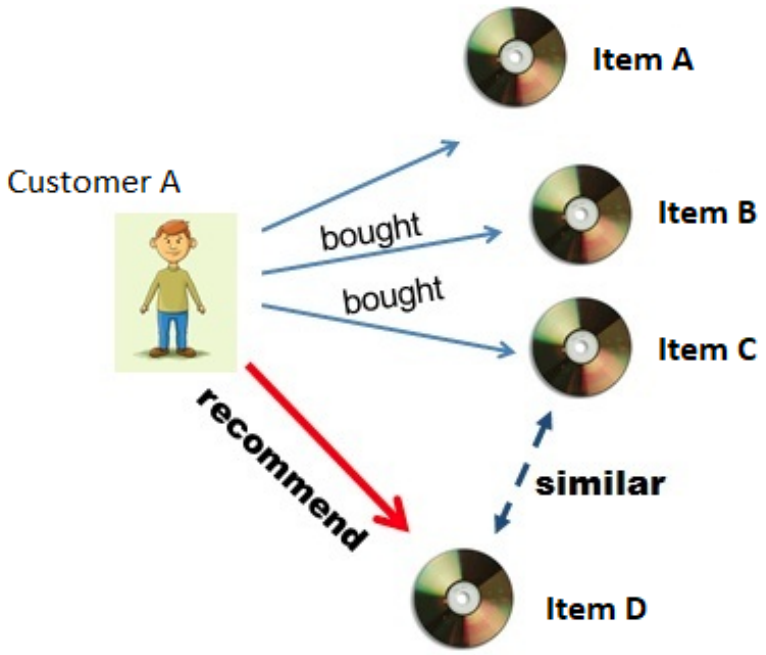

Figure 4.1: Item-based collaborative filtering visualization [20]

similar to 'C', therefore, there is a high likelihood that customer A will also like item 'D' because it is similar to one he has already liked. Hence, we will suggest the item 'D' to customer A.

Collaborative filtering filters information by using the interactions and data collected by the system from other users. It's based on the idea that people who agreed in their evaluation of certain items are likely to agree again in the future [23]. The concept is simple: when we want to find a new movie to watch we'll often ask our friends for recommendations. Naturally, we have greater trust in the recommendations from friends who share tastes similar to our own. Collaborative-filtering systems focus on the relationship between users and items. The similarity of items is determined by the similarity of the ratings of those items by the users who have rated both items. So at its core IBCF is all about finding items similar to the ones user has already liked. To understand this we need to first understand the

intuition behind the process, which will assist us in comprehending the mathematics behind the IBCF recommendation filtering.

4.2 Finding Similarity Between Items

Suppose we have a table of customers and their ratings for items as shown in Fig. 4.2. A rating of five is associated with a high liking of the corresponding item.

Customer	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
Customer 1	2	5	2		4	4
Customer 2	1			2	4	2
Customer 3				5	2	
Customer 4	2	3				1
Customer 5		3	4	1	4	1
Customer 6	3	4	4	4		
Customer 7			3	1	1	4
Customer 8	2				3	2
Customer 9	4	1		4	2	4
Customer 10	4	2		5	3	
Customer 11	5	1	5			5
Customer 12	2	5	2			
Customer 13	4			5	4	
Customer 14	3		3		2	
Customer 15	2	4			5	5

Figure 4.2: Rating data sample for IBCF

Let's pick two items, item 1 and item 2, for which we calculate the similarity i.e. how much these two items are comparable to one another in terms of their likeness by users. To compute this, we first multiply ratings of both items with each other and then sum the result. We call this value **A** as demonstrated in Fig. 4.3.

Next, we compute the square of all item 1 ratings, sum them and then take the square root to obtain a value, denoted as b_1. Similarly, we compute a square root value b_2 for item 2. The $b_1 \times b_2$ product results in value **B** shown in Fig. 4.4. Then,

the division **A**/**B** produces a score of 0.64995277 that indicates similarity between item 1 and item 2 (Fig. 4.5).

Customer	Item 1	Item 2	Product of Item Ratings
Customer 1	2	5	10
Customer 2	1		0
Customer 3			0
Customer 4	2	3	6
Customer 5		3	0
Customer 6	3	4	12
Customer 7			0
Customer 8	2		0
Customer 9	4	1	4
Customer 10	4	2	8
Customer 11	5	1	5
Customer 12	2	5	10
Customer 13	4		0
Customer 14	3		0
Customer 15	2	4	8
Customer 16			0
Customer 17	2		0
Customer 18	4	5	20
Customer 19	2	1	2
stomer 20	2	2	4
		A =	89

Product of Ratings

Sum of Product Ratings

Figure 4.3: Product of ratings of two items

1. Square Ratings

2. Sum of values from 1

Customer	Item 1	Item 2	Product of Movie Ratings	Square of Item 1	Square of Item 2
Customer 1	2	5	10	4	25
Customer 2	1		0	1	0
Customer 3	0		0	0	0
Customer 4	2	3	6	4	9
Customer 5		3	0	0	9
Customer 6	3	4	12	9	16
Customer 7	0		0	0	0
Customer 8	2		0	4	0
Customer 9	4	1	4	16	1
Customer 10	4	2	8	16	4
Customer 11	5	1	5	25	1
Customer 12	2	5	10	4	25
Customer 13	4		0	16	0
Customer 14	3		0	9	0
Customer 15	2	4	8	4	16
Customer 16	0		0	0	0
Customer 17	2		0	4	0
Customer 18	4	5	20	16	25
Customer 19	2	1	2	4	1
Customer 20	2	2	4	4	4
		A = 89		B = 137.9855065	

Sum of Item 1 Ratings	Sum of Item 2 Ratings
140	136
Square Root of Sum of Item 1 Ratings	Square Root of Sum of Item 2 Ratings
11.83215957	11.66190379

3. Square root the result of 2

4. Product of values from 3

Figure 4.4: Square root of sum of squared item ratings

Customer	Item 1	Item 2	Product of Item1*Item2 Ratings	Square of Item 1	Square of Item 2
Customer 1	2	5	10	4	25
Customer 2	1		0	1	0
Customer 3	0		0	0	0
Customer 4	2	3	6	4	9
Customer 5		3	0	0	9
Customer 6	3	4	12	9	16
Customer 7	0		0	0	0
Customer 8	2		0	4	0
Customer 9	4	1	4	16	1
Customer 10	4	2	8	16	4
Customer 11	5	1	5	25	1
Customer 12	2	5	10	4	25
Customer 13	4		0	16	0
Customer 14	3		0	9	0
Customer 15	2	4	8	4	16
Customer 16	0		0	0	0
Customer 17	2		0	4	0
Customer 18	4	5	20	16	25
Customer 19	2	1	2	4	1
Customer 20	2	2	4	4	4
		A = 89		B = 137.9855065	

Sum of Item 1 Ratings	Sum of Item 2 Ratings
140	136

Square Root of Sum of Item 1 Ratings	Square Root of Sum of Item 2 Ratings
11.83215957	11.66190379

Similarity between items 1 and 2

Similarity (Items 1 and 2) = 0.644995277

Figure 4.5: The similarity between item 1 and 2

Repeating the above process for all pairs of items will result in a matrix of similarities between each item. Such an adjusted similarity matrix is shown in Table 4.6. This similarity matrix is symmetric along the first diagonal as $similarity(i, j) = similarity(j, i)$ for any items i and j.

Similarity	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
Item 1	0.12025	-0.0578	0.05114	0.0309	-0.0734	0.02223
Item 2	-0.0578	0.19459	-0.0922	0.00065	-0.0036	-0.0668
Item 3	0.05114	-0.0922	0.17183	-0.0522	0.00341	0.07474
Item 4	0.0309	0.00065	-0.0522	0.17024	-0.0269	-0.0002
Item 5	-0.0734	-0.0036	0.00341	-0.0269	0.12296	0.01425
Item 6	0.02223	-0.0668	0.07474	-0.0002	0.01425	0.21202

Figure 4.6: Adjusted similarity matrix

Definition 4.1 Cosine similarity equation: *The similarity between items i and j is defined in Equation 4.1, with r, X, N defined in Fig. 4.7.*

$$similarity(i, j) = \frac{\sum_{X=1}^{N} r_{(X,i)} * r_{(X,j)}}{\sqrt{\sum_{X=1}^{N} r_{(X,i)}^2} \sqrt{\sum_{X=1}^{N} r_{(X,j)}^2}} \tag{4.1}$$

Equation 4.7 illustrates how the above process is depicted in mathematical form. The details of cosine similarity equation are labeled. Starting from label 1 (blue on left), illustrating that in order to calculate the similarity between an item i and an item j, one multiplies all ratings of item i and j given by customers and sum them. Then, divides the result with the product of square root of the sum of squared ratings of individual items given by customer X. Iterating above for all the items will result in a list of values that will indicate how close other items are to our main item i. In other terms, this process produces similarity scores between the item in row i and all other items. This method is also known as cosine similarity, which helps calculate how close two vectors are to one another.

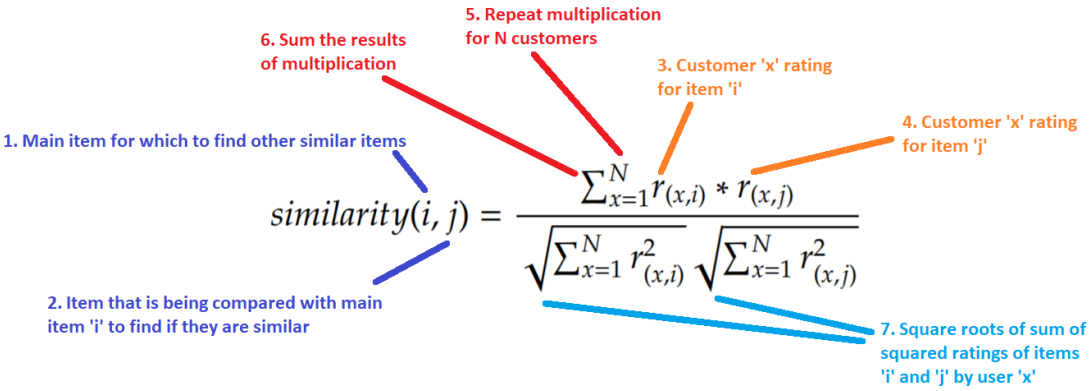

Figure 4.7: Cosine similarity equation, details labeled

The cosine similarity uses $cos(\theta)$ to measure the distance between two items (representing by two vectors, as shown in Fig. 4.8), so as $cos(\theta)$ increases, θ decreases. Therefore, if the similarity value is closer to 1, we have θ closer to 0, and higher similarity between two items. In other words, these two items are likely to be rated similarly.

Cosine similarity works but it does not take into account the optimistic behavior of customers. Different customers can rate the same item differently depending

Figure 4.8: Cosine similarity visualization

upon how optimistic they are. On the scale of 5, one could rate an item 5 while another could rate 3 even though they both very much liked the item. To account for this we make a small change to our similarity formula. Subtracting average rating of customer X, denoted as $\bar{r}_x$ in Equation 4.2, normalizes ratings to the same scale and helps overcome optimism issues.

Definition 4.2 ADJUSTED COSINE SIMILARITY EQUATION: *Adjusted cosine similarity equation between items i and j is defined in Equation 4.2, where $\bar{r}_X$ is the average rating of customer X.*

$$similarity(i,j) = \frac{\sum_{X=1}^{N}(r_{(X,i)} - \bar{r}_X) * (r_{(X,j)} - \bar{r}_X)}{\sqrt{\sum_{X=1}^{N} r_{(X,i)}^2}\sqrt{\sum_{X=1}^{N} r_{(X,j)}^2}} \tag{4.2}$$

There is also a similar method where instead of subtracting the user's average rating, we subtract the item's average rating. This helps to understand how much given user ratings deviate from average item rating. This technique is known as

Pearson similarity. Both Cosine and Pearson are widely used methods to compute similarities [23].

Applying adjusted cosine similarity equation on ratings for items in the example above produces the table or matrix shown in Table 4.6. The table shows that the most similar pair of items is item 3 and item 6, with a similarity score of 0.07474. In comparing the similarity between pair of items, we do not include the similarity score on the first diagonal because it is the similarity between item i and itself, hence, it always has the highest score within a row and a column. Furthermore, the similarity on the diagonal is not 1 although they are the same pair of items because the score has been adjusted based on the average ratings that all customers gave.

4.2.1 Neighborhood Selection

The number of nearest neighbors to be selected and the criteria used for this selection also affect the recommender system's performance. Due to memory constraints, it is not possible to store all the similarities between each pair of users or items in large recommender systems. It is also unnecessary to do so, as in the predictions only the most significant of these values are used. We must therefore reduce the number of similarity weights to be stored and limit the number of candidate neighbors to be taken into account in the predictions. Two common techniques for doing so are Top-N filtering and Threshold filtering [10].

Definition 4.3 Top-N Filtering: *This approach keeps for each user or item only a list of N nearest-neighbors and their respective weights of similarity.*

Storing an unnecessarily large value of N costs extra space and reduces predicting speed as well. On the other hand, selecting a too-small value for N may reduce the accuracy of the recommendation.

Definition 4.4 Threshold Filtering: *Instead of maintaining a fixed number of neighbours, this approach keeps all neighbors whose similarity weight exceeds a predetermined threshold.*

This filtering method is more flexible than Top-N filtering, as only the most appropriate neighbors are preserved. The threshold value must also be carefully determined as it has a huge impact on the recommender system's quality.

In this project, in order to keep the algorithm simple to implement for experimenting and without having to cross-validate for the most appropriate threshold, we Top-N Filtering approach with the top 20% similar items to a product of interest. Note that we already subset customers who bought more than 50 items.

4.3 Calculate Recommendation Scoring

A table of similar items is a half done job. We know which items are comparable but we have yet to recommend to customers from the list of similar items that they might want to buy. The next step is to combine our similarity matrix with customers' past history of rated items to generate a recommendation. This is easily achieved by applying IBCF's equation.

Definition 4.5 Item-based Recommendation System Equation: *The customer X's score for item i is calculated as*

$$score(x, i) = \frac{\sum_j^J similarity(i, j) * (r_{(x,j)} - \bar{r}_j)}{\sum_j^J similarity(i, j)} + \bar{r}_i \tag{4.3}$$

Where:

- **x** = customer for which we are generating recommendation
- **i** = item in consideration

- **score(x,i)** = generates a score that will indicate how strong a recommendation of item i is to customer X
- $j \in J$ = items similar to main item **i**
- $\bar{r}_j$ = average rating of item j

Using Equation 4.5 results in the predicted score matrix for each customer as shown in Fig. 4.9. Scores denoted by -1 represent the items that were already rated by the customer, and we want to distinguish them with those needed to be predicted. Note that sometimes there are negative predicted scores such as customer 3's item 3. This might happen as the most similar items to item 3 have negative similarity scores, and the average rating $\bar{r}_3$ is not high enough to cancel out those negative values. The vice versa also may happen, when the similarity scores of other items are close to 1 and the average rating of that item itself is already high, it is possible that the predicted rating exceeds 5.

From Fig. 4.9, we can conclude that customers 2 and 9 might enjoy item 3, while customer 6 might enjoy item 6.

4.4 Applying Item-Based Collaborative Filtering to Amazon Review Dataset

In this dataset, if we subset customers that bought 50 items or more, there are 832 distinct customers and 35,447 unique products. Since the similarity matrix is a $n \times n$ matrix, where n denotes the number of items, the complexity of generating similarity matrix is $O(n^2)$ and normal computer's memory will be overflowed when executing this program.

For reducing the complexity of the original dataset to fit in the initial baseline model, we select customers who reviewed over 50 products and choose only the

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
Customer 1	-1	-1	-1	1.36042	-1	-1
Customer 2	-1	1.471487	3.45399	-1	-1	-1
Customer 3	1.093284	0.999974	-0.02865	-1	-1	1.159053
Customer 4	-1	-1	1.44793	2.331391	0.513636	-1
Customer 5	0.890072	-1	-1	-1	-1	-1
Customer 6	-1	-1	-1	-1	2.598063	4.064554
Customer 7	1.440993	0.832511	-1	-1	-1	-1
Customer 8	-1	0.051213	0.001753	1.102248	-1	-1
Customer 9	-1	-1	4.197146	-1	-1	-1
Customer 10	-1	-1	2.628469	-1	-1	0.455542
Customer 11	-1	-1	-1	1.784869	1.870011	-1
Customer 12	-1	-1	-1	1.728419	1.68421	0.5871
Customer 13	-1	1.385722	1.217089	-1	-1	1.484091
Customer 14	-1	2.744736	-1	-0.49217	-1	2.671612
Customer 15	-1	-1	3.685631	1.277735	-1	-1

Figure 4.9: Predicted score matrix

items that were rated more than 30 times among those customers. By doing this data manipulation, we might lose some of the customer preference information. However, there are items in the sub-dataset that were bought only by one specific customer. Thus, it is difficult to find similar items and knowing the rating score for that rarely bought item is not valuable in understanding customer's general favor. Eventually, the sub-dataset's complexity and dimension is significantly decreased from 832 customers and 35, 447 products to 404 and 35 frequently visiting customers and rated items, respectively.

4.4.1 Results

The predicted score matrix is computed and reported in Fig. 4.10, where the first column indicates a unique customer ID, and the other columns represent products. Each element in this matrix is the predicted rating that a customer would give if we recommend it. Fig. 4.11 shows customer *b'10011509''*s history of purchase and top five items that the program would recommend.

customer_id	b'B009HEZ03W'	b'B00COCD6TO'	b'B00CUJMDNG'	b'B00DPNCEY4'	b'B00DUHACEE'	b'B00EPPXRCE'	b'B00ESHDGOI'	b'B00GM477G8'	b'B00GPUH
b'10009912'	1.355323	1.052225	-0.047924	-0.039897	-1.000000	0.899865	-1.000000	1.536789	1.474
b'10011509'	0.225780	0.186556	2.508418	2.493798	0.253340	0.175020	0.156661	0.192310	0.185
b'10114275'	-0.174220	-0.213444	-1.000000	-1.000000	-0.146660	-0.224980	-0.243339	-0.207690	-0.214
b'10144173'	-0.774220	-0.813444	-0.356108	-0.231095	-0.746660	-0.824980	-0.843339	-0.807690	-0.814
b'10223755'	1.655979	0.186556	0.118742	0.126769	0.253340	1.532278	-1.000000	0.692296	0.632
...	...	...	...	...	...	...	...	...	
b'8846806'	0.225780	0.186556	-1.000000	0.973037	0.253340	0.175020	0.156661	0.192310	0.185
b'901585'	-0.107553	-0.146777	-1.000000	0.470451	-0.079993	-0.158314	0.315756	-0.141024	-0.148
b'9295389'	0.225780	0.186556	0.118742	0.126769	0.253340	0.175020	0.156661	0.192310	0.185
b'957506'	0.225780	0.186556	0.118742	0.126769	0.253340	0.175020	0.156661	0.192310	0.185
b'9953180'	-0.024220	-0.063444	0.713678	0.858006	0.003340	-0.074980	-0.093339	-0.057690	-0.064

Figure 4.10: Predicted score matrix for Amazon dataset

```
Customer b'10011509' bought:
      product_id                                      product_title  \
0  b'B00NNTBF22'  b'PRO-15 Advanced Strength Probiotics: 3x the ...
1  b'B00OG5O37Q'  b'Immune: Hyperbiotics Daily Immune & Wellness...
2  b'B00RH5I8U0'  b'Green Tea Extract Supplement with EGCG for W...
3  b'B00RH5K26I'  b'Digestive Enzymes Plus Prebiotics & Probioti...

            product_category rated_score
0  b'Health & Personal Care'           5
1  b'Health & Personal Care'           5
2  b'Health & Personal Care'           5
3  b'Health & Personal Care'           5
Recommended items:
      product_id                                      product_title  \
0  b'B00CUJMDNG'  b'Natizo Stainless Steel Measuring Spoon Set -...
1  b'B00DPNCEY4'  b'Liquid Vitamin D Drops - 2oz D3 100 Iu Per D...
2  b'B00KFETGW4'  b"Nature's Fruit ThermoCleanse (30 Servings, 6...
3  b'B00LPR2ZAU'  b'Athelas Neutraceuticals Natural Triple Stren...
4  b'B00ONH0S8U'  b'Hair Growth Vitamins Supplement - 5000 mcg B...

            product_category  recommended_score
0                b'Kitchen'           2.508418
1  b'Health & Personal Care'           2.493798
2  b'Health & Personal Care'           2.848355
3  b'Health & Personal Care'           2.564384
4  b'Health & Personal Care'           3.545424
```

Figure 4.11: Top five predicted rating for customer *b'10011509'*

Previously, customer *b'10011509'* bought probiotics, daily immune hyperbiotics, green tea extract supplement, and digestive enzymes and is very satisfied with those product, rating them all 5/5. Based on this history of purchase, we guess that the customer is having some digestive problems (supported by probiotics) or trying to lose weight (tea extract's EGCG is a powerful compound that may benefit health by

reducing inflammation, aiding weight loss, and preventing certain chronic diseases) [2]. In this situation, IBCF program recommends other Health & Personal Care products that customer *b'10011509'* might be interested in, including vitamin D, fruit's extract, and hair growth supplement, etc. These supplements are often rated similarly to the products that the customer bought, and in practice, they can help in compensating negative side effects of using digestive products over a long time. Therefore, this recommendation is reasonable in terms of real-life circumstance.

4.5 Usefulness of Item-based Collaborative Filtering

4.5.1 Company Level

A huge plus is that companies with online shopping facilities would benefit from ratings provided on relatively new items made by possibly new vendors, especially since some such items would be imperative during this pandemic, e.g. face masks. While N95 masks have been recommended by health organizations, it is found that they are now more prevalent among medical professionals mainly due to the scarcity of these masks. Instead *washable cloth masks* and *disposable surgical masks* are more in use by the general public. Also, since N95 masks are not comfortable to wear for too long, it is observed that almost equally compatible KN95 masks are more common and are recommended for longtime daily use due to their safety, comfort and availability. In fact, KN95 masks come with descriptions such as "perfect for office".

Likewise, the correlations between such pertinent items as captured through item-based collaborative filtering are crucial here, especially with respect to the brands of specific products and their relationships that would provide guidance in online marketing. Customers would be delighted that they do not have to spend too much time on browsing and searching, and would thus be likely to buy more than

one item if similar ones are displayed via the results of item-based collaborative filtering. Also, they would be inclined to revisit the concerned site due to liking item-based recommendations therein, thus possibly buying more items and bringing the company more revenue.

4.5.2 Customer Level

Nowadays people prefer to have personalized recommendations rather than general ones. For example, if the customer is a healthcare professional (or is shopping on behalf of one through a hospital), it would be beneficial to get ratings on items such as full PPE (personal protective equipment) and different brands of gloves from other healthcare professionals who have bought similar items online. Having similar items recommended based on user ratings would imply that the recommendations would be geared towards that customer's likings. By having such recommendations, a customer would easily be able to compare and browse more such products without having to search again. Overall, the customer would enjoy the shopping experience and would also save valuable time, much needed during this pandemic and its aftermath.

4.6 Limitations and Comparisons of User-based and Item-based Filtering

There are some problems which limit the application of neighborhood-based techniques, especially in large-scale systems like Amazon or Netflix. Three characteristics of the system should be considered in order to choose when to use a user-based or item-based neighborhood approach.

1. **The ratio between the number of users and items:** The efficiency of recommender systems is also determined by the number of users to items ratio. The neighborhood-based algorithms can be divided into two primary phases in terms of computational cost: determining the nearest neighborhood and estimating the rating. In the user-based filtering approach, determining a user's closest neighbors necessitates computing all other users' similarity weights. User-based algorithms' time and memory needs scale linearly with the number of users and ratings. Thus, on platforms like Amazon.com, where the number of users much outnumbers the number of items, providing just one recommendation would be prohibitively expensive, while scanning all user ratings would use a significant amount of resources.

   In contrast, item-based solutions are less effective than user-based methods in systems where the number of items exceeds the number of users, such as an online news website, because they are more prone to suffer from the new-item problem. That is, an item must be co-rated by a large number of users before it can be recommended by the algorithm. An item can be recommended as soon as it is rated by a single user using user-based collaborative filtering [13].

2. **Stability of data:** The volatility of users and system items also influences our decision between these two ways. An item-based method may be preferred in systems where the list of available items is relatively static in comparison to the system's users, because the item similarity weights could then be computed at infrequent time intervals while still being able to recommend items to new users. User-based techniques may show to be more stable in applications where the list of available items is continually changing, such as an online article recommender [10].

3. **Sparsity of data:** In fact, the rating data is frequently scanty because most

users only rate a handful of the goods. User-based techniques have a hard time making accurate predictions because pairings of users with few co-ratings are prone to skewed correlations. If two users have only a few co-rated products in common, the similarity weight between them is likely to be 1. Such skewed neighbors can completely dominate a user's neighborhood, resulting in an incorrect prediction [25].

CHAPTER 5

Sentiment Classification and Analysis

In this chapter, we give a brief overview of techniques that model sentiment in document or reviews and we point out a future application of sentiment analysis to our Amazon product reviews.

5.1 Introduction

E-commerce is becoming more and more popular as e-commerce sites allow users to leave reviews on different products. Millions of reviews are generated by customers every day, making it difficult for product manufacturers to keep track of customer opinions about their products. Therefore, it is important to classify such large and complex data to get useful insights from a large data set. The classification method is the way to solve these problems and entails, classifying data into groups or classes based on common characteristics [19].

Sentiment analysis, also known as opinion mining, is a natural language processing (NLP) problem that aims to identify and extract subjective information from textual sources. The purpose of sentiment classification is to analyze the user's written reviews and categorize them into positive or negative opinion, so that the system does not need to fully understand the semantics of every sentence or document [18].

Although sentiment analysis originated in computer science, in recent years it has spread to management and the social sciences due to its importance to business and society at large. Thus, the study of emotion analysis not only advances the field of NLP but also advances research in management science, political science, and economics, as these fields are all interested in opinions of consumers and the public. It is therefore not difficult to imagine that emotion analysis using social media and e-commerce could profoundly change the direction of research and practice in these fields. A common concern for organizations is the ability to automate the classification process of sentiments when large datasets are used [18].

However, this is not done simply by labeling words as positive or negative. Sorting out words and phrases with prior positive or negative polarity does not always work. For example, the word “amazing” has a prior positive polarity, but if it comes with a negation word like "not", the context can completely change. Moreover, the word "unpredictable video camera" has a negative meaning to that camera while "unpredictable experience" is considered positive experience for tourists [28].

Sentiment classification has been attempted in different fields such as movie reviews, travel destination reviews and product reviews. Lexicon based methods and machine learning methods are two main approaches that are usually used for sentiment classification [18].

5.2 Sentiment classification using Machine learning methods

Machine learning tries to develop an algorithm that uses example data to optimize the system’s performance. There are two primary steps in the sentiment analysis solution provided by machine learning. The first stage is to "learn" the model using the training data, and the second step is to use the trained model to classify

previously unseen data [15]. Machine learning algorithms can be classified in different categories:

1. **Supervised Learning** The process by which the algorithm learns from the training data can be compared to a teacher guiding the students' learning. The supervisor is educating the algorithm what conclusions it should produce as an output in some way. As a result, both input and output data are provided. It is also necessary for the training data to be labeled. The output will be more precise if the classifier receives more labeled data. The purpose of this method is for the algorithm to accurately predict output for new input data. The supervisor can direct the algorithm back to the correct path if the output differs significantly from the expected outcome. As long as labeled data is available, supervised learning works properly. There are however some challenges involved when working with supervised learning. This means that if the machine encounters data it has not seen before, it will either classify it incorrectly or eliminate it because it hasn't "learned" how to identify it [6].

2. **Unsupervised Learning** In contrast to supervised learning, unsupervised learning is taught on unlabeled data with no corresponding output. The algorithm should be able to figure out the dataset's underlying structure on its own. This implies it must find comparable patterns in the data in order to decide the output without having the correct answers. Clustering is one of the most essential method in unsupervised learning challenges, which is the process of recognizing similar groups of data in a dataset. It is usually the sentiment words and phrases that are utilized for unsupervised sentiment classification. This means that a review's classification is dependent on the average semantic orientation of the sentences in the review. This is understandable given that sentiment terms are frequently the most important factor in sentiment classification [4].

3. **Semi-supervised Learning** Semi-supervised learning, which combines the advantages of both supervised and unsupervised learning, refers to problems in which only a small portion of the training data set is labeled and the remainder is left unlabeled. This is useful when obtaining data is cheap but labeling it is time consuming and costly. This technique is effective in theory and practice since having a large amount of unlabeled data during the training process tends to improve the accuracy of the final model while requiring much less time and money to build [6].

5.3 Super Vector Machines

Support vector machines (SVM) are a type of supervised learning techniques for handling sentiment classification problems. This method uses a decision plane to place labeled training data, and then an algorithm to generate an optimum hyperplane that divides the data into groups or classes. A simple linear SVM classifier works by making a straight line between two classes. That means all of the data points on one side of the line will represent a category and the data points on the other side of the line will be put into a different category. This means there can be an infinite number of lines to choose from.

Fig. 5.1 presents a simple classification problem and also provides three possible different separating hyperplanes. The best hyperplane is the one that separates the classes by the greatest margin. This is accomplished by selecting a hyperplane with the greatest distance from the nearest data on each class [4]. In the example, H1 does not separate the classes. H2 does, but only with a small margin. H3 separates them with the maximum margin. The two data points connected to H3 are called support vectors because they are used in computing the equation of the hyperplane H3.

Regarding Amazon Review dataset, it is reasonable to label reviews into three categories: positive (rated score of four or five), neutral (rated score of three), and negative (rated score of one or two). Thus, there should be two hyperplanes resulting from SVM model. Furthermore, SVM performs effectively in a dataset where the number of dimensions is greater than the number of samples, but does not perform well when we have large data set because the required training time is exceptionally long. As a result, the original dataset needs to be subset in a rational way so that we do not lose much information from the original dataset as well as keep the sub-dataset small enough for SVM to be executable.

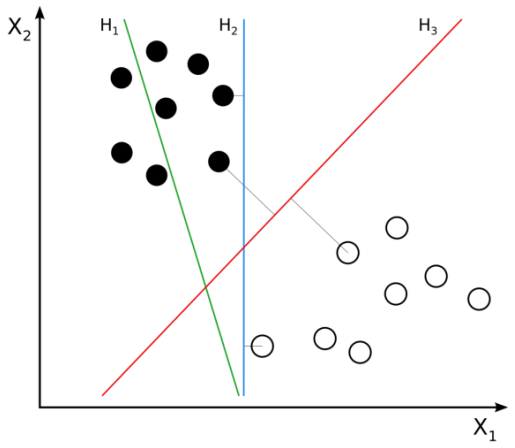

Figure 5.1: H1 does not separate the classes; H2 does, but only with a small margin; H3 separates them with the maximum margin [4]

5.4 K-Means Clustering

K-means clustering is an unsupervised iterative learning approach that allows us to classify data points that do not have available differentiating label. This technique compares the similarity of the data points and assigns them to different clusters as a result. It aims to divide the data set into k unique clusters, each of which is unrelated

to the others and where the intracluster data points are as similar as possible (as shown in Fig. 5.2).

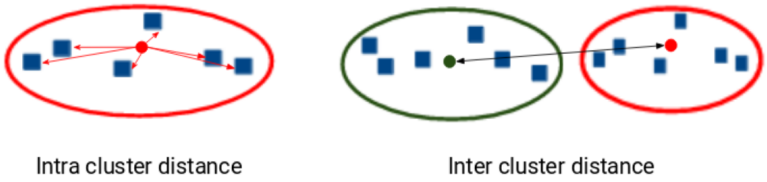

Figure 5.2: Distance between data points in (a) Intra cluster, (b) Inter cluster

The K-means algorithm starts by generating k centroids at random from a data set. These centroids are used as the starting point for a cluster, which expands as the k centroids combine other data points together. To optimize the position of these positioned centroids, this process is repeated iteratively. When the centroids have identified a stable location and their values have not changed after the clustering has been successful, the iteration ends. The term *stage* is indicated by low changes between data points within each cluster. It can also be terminated when a certain number of iterations have been completed. This is frequently done to keep the run time under control.

Fig. 5.3 demonstrates the result of perform K-Means Clustering with $k = 3$ on a dataset containing age and income information. Initially, no label was provided. Each color denotes a cluster after executing the algorithm. As a result, the program grouped the data points based on their distance to each cluster's centroid, shown as three squares in the figure. Although there were no tags for the data points, according to the clusters, it is clear that the data points are classified into high income (green), average income (red), and low income (black).

This clustering technique requires us to specify the number of clusters the data set should be divided into. In other words, it requires us to have prior knowledge of the data being analyzed, or forces us to make assumptions about it. Since this is an iterative and randomized algorithm, reproducing results are not feasible and

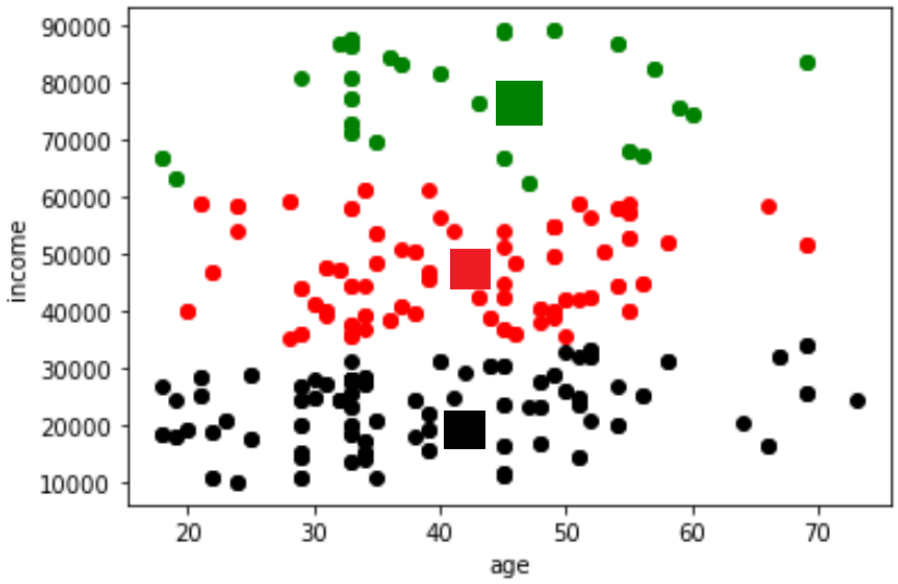

Figure 5.3: Three clusters of income data points, each square indicates a centroid of each cluster

hence, not always reliable. To overcome this, a common practice implemented is the Elbow Method. This enables the user to determine the optimal value of k in the K-means clustering algorithm. The basic principle of the Elbow Method is to run the K-means clustering algorithm on the data set for a specified range of k values. For every value of k, the inertia is calculated. This value is a measure of the sum of the squared distances of data points to their centroid. For each value of k, the inertia value is plotted and this line graph tends to form the shape of a human arm. The ideal k value is the value that coincides with the "elbow" of the plotted line. The concept this revolves on is that we desire smaller inertia values. It must be noted that the inertia value is equal to zero when the number of clusters k is equal to the number of data points in the data set. This defeats the purpose of grouping and treats every instance as a group itself. Therefore, the ultimate goal is to have a low inertia value.

Fig. 5.4 illustrates the Elbow Method for the K-Means Clustering above. It is understandable that the overall error decreases as the number of clusters increases, since the distance between the data points and their centroid is closer than that

when less clusters are generated. Therefore, the Elbow Method is necessary to find the optimal k, which is the point where the sum of squared error stopped declining significantly as k increases by one unit. In this example, it can be observed that the error starts to flatten out between four and five clusters, thus, the optimal k is three clusters.

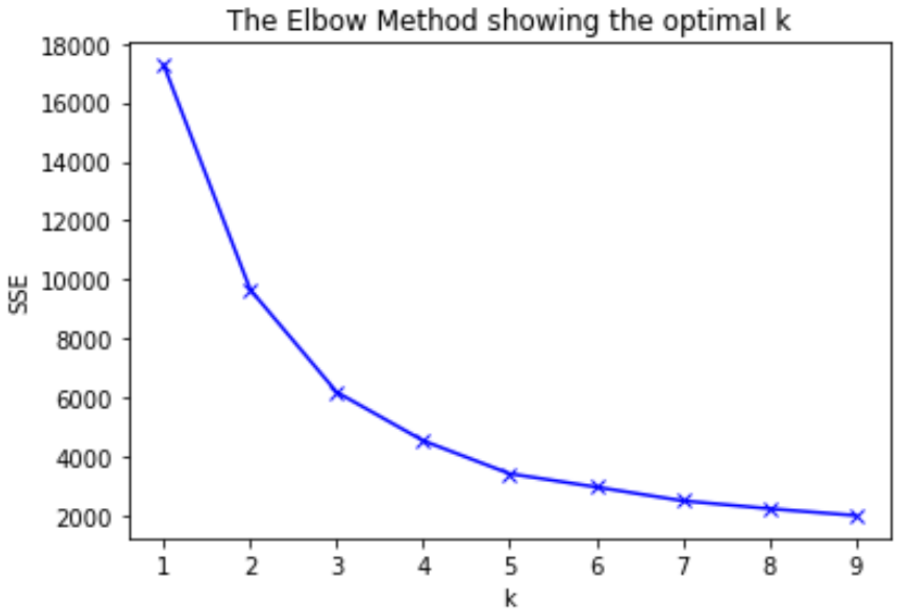

Figure 5.4: Three clusters of income data points, each square indicates a centroid of each cluster

5.5 Sentiment classification using Lexicon based methods

5.5.1 Introduction to Lexicon based method

Lexicon based method is another unsupervised approach, which relies on word and phrase annotation. The motivation of the method is that words and phrases that convey positive or negative sentiment are instrumental for sentiment analysis. To compute a sentiment score for each text, this method uses a dictionary of

sentiment words and phrases. Positive sentiment words such as *beautiful*, *wonderful*, and *amazing* are used to express some desired states or qualities while negative sentiment words such as *bad*, *awful*, and *poor* are used to express some undesired states or qualities. In lexicon-based methods the simplest approach for determining the sentiment of a review document where is to use a count-based approach. If we have a text and a lexical resource containing the positive and negative annotation of words and phrases, we can assign the polarity of the review. This means that if the number of positive words is more than the negative ones the polarity of the review is positive. If there are more negative sentiment words than positive sentiment words, the overall sentiment of the text is then negative.

5.5.2 Bag of Words Model

Tokenization is the process of splitting up a sentence into fragments, where each token is a word. This is also called text segmentation. We should note that this process also converts every string into its lowercase version and rids the dataframe of punctuation. An example of this can be seen in Figure 5.5 which is converted to the entries seen in Figure 5.6. The thought is to create a list of words that are merely present in the sentence being studied. This reduces chances of error by preventing *"this"* or *"This"*, and *"string"* or *"String"* being tokenized into different words, when they convey the same meaning.

Normalizing the tokenized data entries means eliminating ambiguity from the data entry that is being studied. The text is normalized by implementing the techniques called stemming or lemmatization. This algorithm collects all inflected forms of a word in order to break them down to their root dictionary form or lemma. Words are broken down into a part of speech (the categories of word types) by way of the rules of grammar. For example, treating the words *"go"* and *"going"* as two different meanings could be misleading to the overall conclusions formed in a

	user	text
1	User1	This is a text string
2	User2	This is another string entry.

Figure 5.5: Original sentences

	user	word
1	User1	this
1.1	User1	is
1.2	User1	a
1.3	User1	text
1.4	User1	string
2	User2	this
2.1	User2	is
2.2	User2	another
2.3	User2	string
2.4	User2	entry

Figure 5.6: Tokenized words

study. Figure 5.7 shows the result of lemmatization to the words *"going"*, *"gone"*, and *"goes"* to their root *go*.

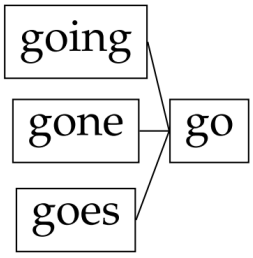

Figure 5.7: Lemmatization

Once the text data is converted into tokens and have been normalized by stemming, the Bag of Words model is implemented. This process helps us to represent a text string in numeric form. Hence, the Bag of Words model is used to preprocess the text by converting it into a vocabulary list which keeps account of the frequency of words in each text document.

A dictionary is initialized as a first step. Then each document (in our case, review) is broken down by tokenizing every word in it. If the word (token) does not exist in the dictionary, it will be added to it in the form of a new column. It should

be noted here that during the first iteration of the algorithm, when the first word is being analyzed, the dictionary is empty. In this manner, the dictionary will consist of all unique words in the data collected. The column headers j will be the tokenized words, and each row i will represent a review or document posted by a user. The $[i, j]$ entry will be the frequency of the j^{th} word appearing in the i^{th} review. This can be demostrated in Tables 5.1 and 5.2. The loss of contextual meaning, where looking at the frequency of phrases does not always convey the exact point the user was attempting to make, is one of the problems with this method. To overcome this, the text data is further refined by implementing the Term Frequency- Inverse Document Frequency statistic, described below in Section 5.5.3.

Customer	Review
Customer 1	This camera is amazing.
Customer 2	This phone is powerful.

Table 5.1: Bag of Words example - original review

Customer	this	camera	is	amazing	phone	powerful
Customer 1	1	1	1	1	0	0
Customer 2	1	0	0	1	1	1

Table 5.2: Tokenized reviews - corpus vocabulary and the frequency of each word in the document

5.5.3 Term Frequency - Inverse Document Frequency

Each review is broken down into separate words and documented in the 'bag-of-words' technique. We lose context information about the full sentence as a result of this process, but we end up with a list of words. As a result, a sentence's semantic meaning is partially lost. A numerical statistic called Term Frequency- Inverse Document Frequency (TF-IDF) is used to solve this problem. This assigns a weight to each term in a document, resulting in values indicating the relevance of a word

in a document. The process of calculating this index is two-fold. For each word in a document, the term frequency is first examined.

Definition 5.1 Term Frequency: *Term Frequency (TF) measures how frequently a term occurs in a document. To account for longer sentences having higher frequencies of certain words, compared to shorter sentences, the TF is normalized by dividing the word frequency by the number of words in the document.*

$$TF_{i,j} = \frac{Number\ of\ occurrences\ of\ term\ t_i\ in\ document\ d_j}{Total\ number\ of\ words\ in\ document\ d_j} \tag{5.1}$$

Where:

- t_i = a specific term/word in the review
- d_j = document j, or j^{th} review
- $TF_{i,j}$ = term frequency of term t_i in document d_j

The Inverse Document Frequency (IDF) is calculated once the TF value is known. This following process addresses a critical problem with the TF as a standalone metric, where all terms are compared to a single document (review) and not the entire corpus, or the collection of text documents, being analyzed.

Definition 5.2 Inverse Document Frequency: *Inverse Document Frequency is logarithmically normalized statistic that measures the importance of a term t_i in the document d_j. This ensures that common words (such as "the", "that", and "is") are given smaller weights compared to rarer words which convey more meaning to the document.*

$$IDF_i = log\left(\frac{Number\ of\ documents}{Number\ of\ documents\ containing\ term\ t_i}\right) \tag{5.2}$$

These two metrics combine to form the *TF_IDF* score by evaluating their product.

$$TF\_IDF_{i,j} = TF_{ij} \times IDF_i \tag{5.3}$$

The words with higher TF_IDF score are given greater importance in the text. Let's take the previous two sentences shown in Table 5.1 for example to compute the TF_IDF value of each word.

$$TF_{2,1} = \frac{\text{Frequency of term } t_2 \text{ camera in } d_1}{\text{Total number of terms in } d_1} = \frac{1}{4} = 0.25$$

$$IDF_{2,1} = log\left(\frac{\text{Number of documents}}{\text{Number of documents containing term } t_2}\right) = log\left(\frac{2}{1}\right) = 0.693$$

$$TF\_IDF_{2,1} = 0.25 \times 0.693 = 0.173$$

This means that $t_{2,1}$ has a TF_IDF score of 0.173 in this dataset. Similarly, TF_IDF scores of all other terms are shown in Table 5.3.

Customer	this	camera	is	amazing	phone	powerful
Customer 1	0	0.173	0	0.173	0	0
Customer 2	0	0	0	0	0.173	0.173

Table 5.3: TF_IDF scores of each word in the document example

However, using only sentiment words and phrases for sentiment classification is not enough. Sentiment lexicon for sentiment analysis is necessary but it is not sufficient [17] as explained next.

1. Positive or negative sentiment words may have different interpretation in different domains. For instance, the word "suck" usually has a negative sentiment, but it can also indicate a positive sentiment. For example, "This camera sucks" expresses a negative opinion but "This vacuum cleaner really sucks" is a positive opinion.

2. Sarcastic sentences are usually hard to deal with even if they contain sentiment words. For example, "What a great car! It stopped working in two days." This is a negative opinion even though it contains the word ”good” which is a positive word.

3. It can happen that a phrase or opinion does not have a sentiment word, so it makes it hard for the machine to compute a sentiment score for the opinion. "This washer uses a lot of water" has no sentiment words but it implies a negative opinion about the washer.

Hu and Liu (2004) [14] work is one of the first studies on this method. They proposed a lexicon-based algorithm for sentiment classification. Since they believed that a review usually contains some sentences with negative opinions and some sentences with positive opinions, they performed classification at the sentence level. For each sentence they identify if it is expressing a positive or negative opinion and then a final summary of the review is produced.

The next step of this work left for future work involves modeling these techniques to analyze the written reviews that accompany the Amazon data set. One interesting research question is to apply sentiment analysis between ratings of five and those of four, for various countries so that we better understand different cultured ways of expressing sentiment.

CHAPTER 6

Conclusions and Future Work

This study presents exploratory research that uses and illustrates intelligent data mining approaches such as association rules and item-based collaborative filtering in the context of an e-commerce baseline recommender system. This study focuses on online Kitchen, Home Improvements, and Health & Personal Care purchases and uses Amazon review data as its dataset for exploratory research. By examining a user's purchase history, the association rule mining technique, which uses the Apriori algorithm, helps us detect patterns of products frequently purchased together. Based on a specific product of interest, measured values from users' ratings of comparable / related products, the item-based collaborative filtering technique suggests highly associated products. Based on the outputs of association rules and item-based collaborative filtering, the baseline recommender system developed in this work makes interesting suggestions to online buyers. Because of the way techniques have been adapted and implemented in this system, it is quite scalable; thus, the techniques integrated into this approach can be successfully deployed on other larger datasets with appropriate adjustments and more powerful computer.

Some future work includes experimenting with content-based sentiment analysis for opinion mining of users' reactions to products, which entails the polarity classification of emotions in text. Since the dataset used in this work includes written

reviews, the classifications of the attitudes towards the products can potentially be useful to improve recommendations. Furthermore, scalable Association Rules can be extended so that it matches pairs of item bundles instead of single item as implemented in this project. IBCF can also be used in larger sub-dataset for more insightful discovery.

References

1. itertools - functions creating iterators for efficient looping - python 3.10.3 documentation. URL https://docs.python.org/3/library/itertools.html. 23

2. Egcg (epigallocatechin gallate): Benefits, dosage, and safety, Apr 2019. URL https://www.healthline.com/nutrition/egcg-epigallocatechin-gallate. 46

3. Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. *Proceedings of the 20th International Conference on Very Large Data Bases*, 1215:487–499, 1994. 13, 20, 21

4. Richard Berk. *Statistical Learning From a Regression Perspective*. Springer, 2016. xvi, 53, 54, 55

5. J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. Recommender systems survey. *Knowledge-Based Systems*, 46:109–132, 2013. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2013.03.012. URL https://www.sciencedirect.com/science/article/pii/S0950705113001044. 5

6. Jason Brownlee. Supervised and unsupervised machine learning algorithms, Mar 2016. URL https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/. 53, 54

7. Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12, 11 2002. doi: 10.1023/A:1021240730564. 2

8. Patricio Cerda and Gaël Varoquaux. Encoding high-cardinality string categorical variables. *IEEE Transactions on Knowledge and Data Engineering*, 34:1164–1176, 2022. 16

9. Brian Connolly. Top amazon product categories, Aug 2021. URL http://www.junglescout.com/blog/amazon-product-categories/. 9

10. Christian Desrosiers and George Karypis. *A Comprehensive Survey of Neighborhood-Based Recommendation Methods*, pages 107–144. 01 2011. doi: 10.1007/978-0-387-85820-3_4. 41, 48

11. Michael D. Ekstrand, John T. Riedl, and Joseph A. Konstan. Collaborative filtering recommender systems. *Foundations and Trends® in Human–Computer Interaction*, 4(2):81–173, 2011. ISSN 1551-3955. doi: 10.1561/1100000009. URL http://dx.doi.org/10.1561/1100000009. 1, 33

12. Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufhann, 3rd edition, 2012. 4, 20

13. Jon Herlocker, Joseph Konstan, and John Riedl. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Information Retrieval*, 5:287–310, 01 2002. doi: 10.1023/A:1020443909834. 48

14. Minqing Hu and Bing Liu. Mining and summarizing customer reviews. *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177, 2004. 64

15. Jayashri Khairnar and Mayura Kinikar. Machine learning algorithms for opinion mining and sentiment classification. *International Journal of Scientific and Research Publications*, pages 1–6, 2013. 53

16. Haosong Li and Phillip C.-Y. Sheu. A scalable association rule learning heuristic for large datasets. *Journal of Big Data*, 86(8), 2021. 21

17. Bing Liu. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, pages 1–167, 2012. 63

18. Bing Liu. *Sentiment analysis: Mining opinions, sentiments, and emotions*. Cambridge University Press, 2015. 51, 52

19. Priyank Pandey, Manoj Kumar, and Prakhar Srivastava. Classification techniques for big data: A survey. *Computing for Sustainable Global Development (INDIACom), 2016 3rd International Conference*, pages 3625–3629, 2016. 51

20. Muffaddal Qutbuddin. Comprehensive guide on item based recommendation systems, Oct 2020. URL https://towardsdatascience.com/comprehensive-guide-on-item-based-recommendation-systems-d67e40e2b75d. xv, 35

21. Paul Resnick and Hal R. Varian. Recommender systems. *Commun. ACM*, 40: 56–58, 1997. 2

22. Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An Open Architecture for Collaborative Filtering of Netnews. *Proc. ACM 1994 Conf. Computer Supported Cooperative Work*, pages 175–186, 1994.

23. Badrul Sarwar, George Karypis, Joseph A Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. *Proceedings of the 10th International Conference on World Wide Web, WWW 2001*, pages 285–295, April 2001. 35, 41

24. Ben Schafer, Joseph Konstan, and John Riedl. Recommender systems in e-commerce. *1st ACM Conference on Electronic Commerce, Denver, Colorado, United States*, 10 1999. doi: 10.1145/336992.337035. v

25. Ben Schafer, Ben J, Dan Frankowski, Dan, Herlocker, Jon, Shilad, and Shilad Sen. Collaborative filtering recommender systems. 01 2007. 49

26. J. Ben Schafer, Joseph A. Konstan, and John Riedl. E-commerce recommendation applications. *Data Mining and Knowledge Discovery*, 5(2):115–153, 2001.

27. Raj Shashi, Ramesh Dharavath, M. Sreenu, and Krishan K. Sethi. Eafim: efficient apriori-based frequent itemset mining algorithm on spark for big transactional data. *Knowledge and Information Systems*, 62(9):3565–3583, 09 2020. xv, 19, 20

28. Zeenia Singla, Sukhchandan Randhawa, , and Sushma Jain. Statistical and sentiment analysis of consumer product reviews. *Computing, Communication and Networking Technologies (ICCCNT), 2017 8th International Conference*, pages 1–6, 2017. 52

29. Brent Smith and Greg Linden. Two decades of recommender systems at amazon.com. *IEEE Internet Computing*, 21(3):12–18, 2017. doi: 10.1109/MIC.2017.72. 33

30. Shihab Uddin Tareq, Md. Habibullah Noor, and Chinmay Bepery. Framework of dynamic recommendation system for e-shopping. *International Journal of Information Technology*, 12:135–140, 2020. 3

31. Don E. Waldman and Elizabeth J. Jensen. *Industrial Organization: Theory and Practice*. Routledge, 4th edition, 2016. 28

32. Chengqi Zhang and Shichao Zhang. *Association Rule Mining*. Springer, 2002. 11

33. Heng-Ru Zhang, Fan Min, Xu He, and Yuan-Yuan Xu. A hybrid recommender system based on user-recommender interaction. *Mathematical Problems in Engineering*, 2015:1–11, 02 2015. doi: 10.1155/2015/145636. 5