E-Commerce Recommendation System Using Association Rule

Vinh Nghiem To

1 Introduction

All of us today have encountered recommendation systems in some facet of our daily lives. 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 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.

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 [1].

Amazon uses 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 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; and a "related to items you have viewed" list based on items similar to those the given user has already seen. Their real

time recommendations given by item-to-item collaborative filtering are surely powerful e-commerce methods. Our work in this paper is orthogonal to such work and is conducted with the long-term goal of mobile application (app) development in order to enhance the shopping experiences of targeted users.

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 [2]. Among the various data mining techniques, a widespread one for years 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.

2 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, we are releasing 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 that we are using is the Amazon apparel 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 have to 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 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. 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. 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 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 [4]. To ensure the quality of review's ratings, only reviews with 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 bigger datasets and hence is important from a scalability angle.

Thereafter, the summary statistics is retrieved pertaining to the ratings column in the data data-frame. As observed here in Fig. 2, the total count of ratings is 1.937,848 with a mean of 395, standard deviation of 1.49, minimum rating of 1, and maximum rating of 5. Also, more than half of the products have a rating of 5 which is good. As depicted in Fig. 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: Summary of statistics on the *star_rating* column

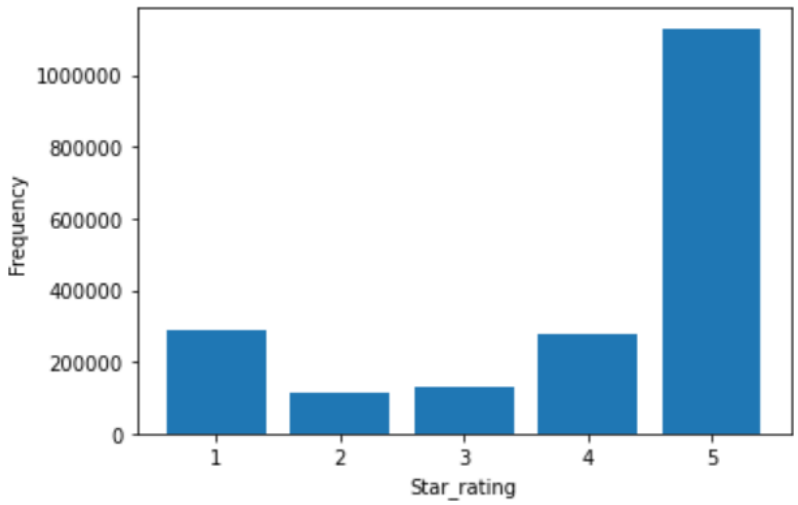

Figure 3: Distribution of ratings histogram

3 Association Rule

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 increase 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 time into account. All items corresponding to a unique customer 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 [6]. 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 comprises of bread, egg, and milk. From such set, we are able to obtain the following antecedents and consequents, denoted as $\{X\} \rightarrow \{Y\}$. 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 2: 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. The following table illustrates five customers shopping in a grocery store and the items that each of them bought.

3.2 Support

The support of Association Rule denoted by *Support*($\{X\}$) is a measure of how frequently an *itemset* occurs among all transactions. The higher value of support indicates that the itemset is more

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: Example of an itemset database

popular to consumers.

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

To illustrate equation (3.1), let consider the itemset *{Bread, Milk}*, which occurs three times in the customers' baskets (in transaction 1, 4, 5) and there are five customers in total. The computation of the support for *{Bread, Milk}* is as follow.

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

An itemset in the database is called frequent if its support is equal or greater than the threshold minimum support given by users. Particularly, 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.

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.

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

For instance, there are two customers buying *{Coke}* out of three who already had *{Bread, Milk}*

in their basket.

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

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. Let introduce an example where this might be a drawback.

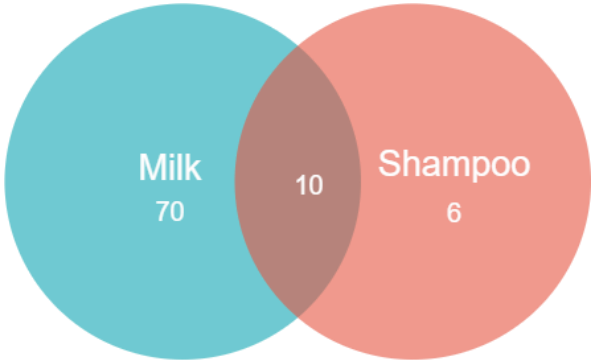

Figure 4: Quantity of milk and shampoo sold in a convenient store

As shown in Fig.4, suppose a store sold a total 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.

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

This is a high confidence value, but intuitively these two products have a weak correlation, or association. 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 problem by controlling the *Support* of consequent while computing the *Confidence* of $\{Y\}$ given $\{X\}$. Mathematically, *Lift* is written as

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

It is implied from Equation 3.7 that the greater the value of *Lift* from 1, 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.

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

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 milk to be bought simultaneously. This helps preventing from *Confidence*'s bias conclusion. The measure of *Lift* supports store managers to decide product organizing on aisle.

4 Association Rule Mining

4.1 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, which has all unique items as columns and each row as a unique customer. Observations with value of 1 represents an item that was bought by the customer while 0 denotes an item was not bought. The following table is an One-Hot Encoded Boolean table created from the itemset database of Table 3.

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 4: One-Hot Encoding table of Table 3

4.2 Generating itemsets from a list of items

The next step in building Association Rule model, we need to generate the *frequent* itemsets from the transactions. For simplicity, suppose a store only sells *Bread, Milk, Coke, Beer*. The size of the possible itemsets vary from one to the number of unique items.

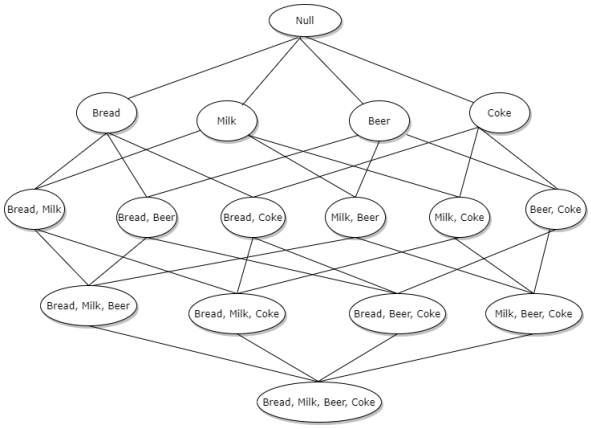

Figure 5: Possible itemsets generated from *{Bread, Milk, Coke, Beer}*

The next step is to compute the *Support* using Equation 3.1 of each itemset and subset those with higher *Support* value than the minimum threshold of *Support* that we set. Such itemsets are called *frequent* itemsets.

4.3 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 any itemset containing beer and coke does not cross the threshold also. Furthermore, this technique generates new itemsets by adding another item to the current frequent itemsets and checking on the Support values repetitively. The Apriori algorithm is shown in Fig.6 below.

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.

4.4 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 [7]. 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

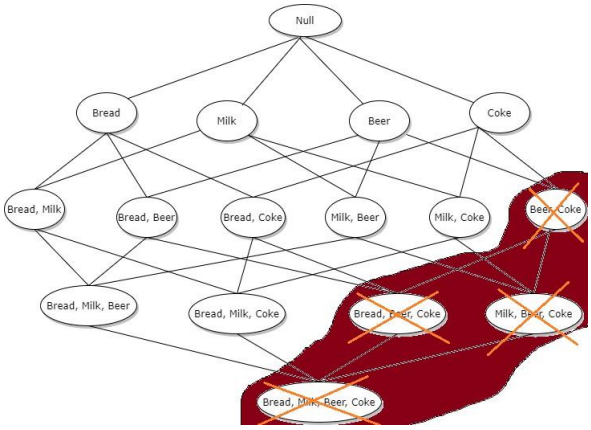

Figure 6: Applying Apriori algorithm to the imtesets generated from *{Bread, Milk, Coke, Beer}*. If itemset *{Beer, Coke}* is not frequent, it is not necessary to take into account all its supersets.

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 [8].

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 [7]) 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 [9].

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.

5 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, splitting the data into multiple subsets to get functions like crosstab and combinations to run on my machine with 8 GB of memory. But even with this approach, my computer could only process about 1800 items before my kernel would crash. To solve this problem, we shall use Python generators.

5.1 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 (eg: 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

5.2 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 replicated it for four times. To make our sample more realistic with a variety of customers buying the same range of goods, we added 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 the following table.

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

Customer 3d	Item 5
Customer 4d	Item 6
Customer 4d	Item 7
Customer 5d	Item 8
Customer 6d	Item 9
Customer 7d	Item 10
Customer 8d	Item 11
Customer 8d	Item 12

Table 5: Testing sample for Association Rules program

The next step of pre-processing data is to group the items bought by customer, which is shown in Table 6.

Customer ID	Items
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 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 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 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 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

Table 6: Transformed data frame from Table 5

5.2.1 itertools.combinations(iterable, r)

The method returns *r* length subsequences of elements from the input *iterable*.

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.

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 (Table 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 Tabe 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*}* (5.1)

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

5.2.2 Demonstration Result

Initially, we measure the *support* value of each item and compare it to our set *min-supp*. Here, we choose *min-supp* to be as low as 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. 7. Equation 5.2 lists the customers and item pairs generated from the sample's Association Rules.

Customers:

Customer 2, Customer 4, Customer 8, Customer 2a, Customer 4a, Customer 8a

Customer 2b, Customer 4b, Customer 8b, Customer 2c, Customer 4c, Customer 8c (5.2)

Customer 2d, Customer 4d, Customer 8d

Item pairs:

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

```
%%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 7: Number of orders, items, and item pairs that meet the requirements

5.3 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).

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 should buy next or by recommending additional products when they are considering a given one. 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 through recommendation systems.

Fig. 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 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. 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 index 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 there is a 0.288 proba-

bility that user also buys *Ubervita Ubersurge Endurance Pre Workout* (based on *confidence*).

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

- The result from this example suggests firms to differentiate their products and recommend to customers goods with similar usage which can complement each other.

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 8: Initial outputs from association rule mining

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 9: Mapping of product IDs to product names

6 Usefulness of Association Rule Mining

6.1 Company

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.

6.2 Customer

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 as 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.

References

[1] S. Tareq, M. Noor and C. Bepery. *Framework of dynamic recommendation system for e-shopping*. International Journal of Information Technology, Vol. 12, 2020, pp. 135140.

[2] J. Han, M. Kamber, and J. Pei, "Data Mining: Concepts and Techniques", 2012, pp. 8

[3] J.B. Schafer, J.A. Konstan, and J. Reidl, *E-Commerce Recommendation Applications*. Data Mining and Knowledge Discovery, Kluwer Academic, 2001, pp. 115-153.

[4] B. Connolly, (2021, August 4). *Top Amazon product categories*. Jungle Scout. Retrieved October 4, 2021, from https://www.junglescout.com/blog/amazon-product-categories/.

[5] P. Resnick et al., *GroupLens: An Open Architecture for Collaborative Filtering of Netnews*. Proc. ACM 1994 Conf. Computer Supported Cooperative Work, ACM Press, 1994, pp. 175-186.

[6] Z. Chengqi, Z. Shichao. *Association Rule Mining*. Springer, 2002. ISBN: 978-3-540-46027-5

[7] R. Agrawal, R. Srikant. *Fast algorithms for mining association rules*. In: Proceedings of 20th international conference very large data bases, vol. 1215, VLDB. 1994, pp. 487–99

[8] J. Han, and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2006.

[9] H. Li, P.CY. Sheu. *A scalable association rule learning heuristic for large datasets*. Journal of Big Data 8, 86 (2021). https://doi.org/10.1186/s40537-021-00473-3