Erasmus University Rotterdam

Erasmus School of Economics

Master Thesis in Data Science and Marketing Analytics

Recommender systems: A comparison between user-based and item-based one-class collaborative filtering models

Author:
K. de Gruiter
Student ID:
413160

Supervisor:
prof. dr. S.I. Birbil
Second assessor:
dr. S.L. Malek

Abstract

This paper examines the performance difference between user- and item-based recommendations, when computed in a One-Class Collaborative Filtering setting. The methods used are user- and item-based collaborative filtering and association rule mining. After computing top-N recommendation lists, the methods are compared based on their mean absolute error, precision, recall and F-measure. The results indicate that, in general, item-based recommendations outperform their user-based counterparts. However, these results are subject to a very skewed item popularity distribution, leading to the fact that we cannot assume a performance gain of using collaborative filtering methods over just recommending the most popular items.

Keywords: Recommender systems, One-class collaborative filtering, Unary purchase data

Thursday 25th March, 2021

Contents

1 Introduction

Nowadays, choosing which product to buy can be a challenging decision. The popularity of online shopping has led to an enormous amount of different products available and customers are continuously comparing different products, brands or retailers in order to maximize their own utility (Zhou et al., 2007). Because of this competitive nature characterizing online shopping, it is very important for retail companies to find ways that build strong relationships with their customers.

One way of enhancing customer relationships is through cross-selling (Kamakura et al., 1991). Cross-selling can be defined as selling complimentary products to customers that already purchased another item before. Some advantages of cross-selling are given by Kamakura (2008), starting with the idea that it is five times more costly to attract new customers than it is to serve existing customers. The next advantage is that cross-selling broadens the scope of the relationship with the customer. Having a broader scope means that from the customer's point of view it will be more costly, both monetary and psychological, to switch to different retailers, which increases the retention rate. In addition, customers buying more different items and widening the scope of the relationship also means a broader range of information available to the firm regarding the preferences of their customers. Because of these benefits of cross-selling, it is important for firms to enhance their cross-selling strategies.

A possible method that strengthens the cross-selling strategy is by using recommender systems. Recommender systems have become increasingly important over the last years, since making accurate predictions about what (potential) customers are interested in, proves to be a very valuable tool.

The systems that compute these recommendations can be divided into two main categories: the Collaborative Filtering (CF) methods and the Content-Based (CB) filtering methods (Park et al., 2012). CF methods make use of the information about previous purchases, ratings or encounters of customers. This information is then compared to a community of customers, from which the behavior of the similar minded customers is used to predict the affinity for products that have not yet been sold to the customer of interest (Bobadilla et al., 2013). CF is considered the most successful approach, but it does have some challenges to conquer. The first challenge is posed by the increase in the amount of information available to use for making recommendations. Traditional collaborative filtering methods, of which the k-nearest neighbor method is the most widespread used, tend to become much slower when the amount of data increases by large amounts (Sarwar et al., 2001). The second challenge lies with producing high

quality recommendations when data becomes sparse. If there is little information about each customer, it can become hard for traditional collaborative filtering methods to consistently make accurate recommendations.

Regarding the second main category, content-based filtering, these methods analyze the content of products, services or encounters a customer has rated. Based on these ratings, content-based filtering methods draw up a user profile based on the characteristics of the ratings and use these profiles to recommend items that are similar to the ones that were positively rated (Park et al., 2012). One of the major pitfalls with these kind of methods is that they tend to give overspecialized recommendations including only items that are very similar to the ones already rated or bought.

The decision as to which method to use for making recommendations is strongly influenced by the characteristics of the data on which the recommendations are based. CF methods mostly rely on the similarities in purchase behaviour, where we expect users to behave similarly in the future if they have shown similar purchase behaviour in the past (Jannach et al., 2010). CB filtering is mostly based on the characteristics of items and the value that users assign to each of these characteristics. It is immediately noticeable that there is a difference in the level of information needed for both methods. Knowing only the purchase history of a set of users can be enough to compute CF recommendations, whereas CB filtering needs more detailed information about the characteristics of the items purchased and how each characteristic was valued in order to compute recommendations.

As obtaining detailed information about preferences of users can prove very challenging for many retailers, this research focuses on making recommendations when only unary purchase history is known. These problems can be categorized as One-Class Collaborative Filtering (OCCF) problems where the data usually consists of unary ratings, depicting a user's action or inaction (Pan et al., 2008). The data that is used, originating from an online grocery store, contains a large number of orders, showing the purchase behaviour of a set of users. The data set is split into training and testing subsets in order to compare several models in terms of performance. The models that are being examined are User-Based Collaborative Filtering (UBCF), Item-Based Collaborative Filtering (IBCF) and Association Rule Mining (ARM). Both UBCF and IBCF use similarity measures to find a nearest neighbourhood, consisting of similar users and items (Jannach et al., 2010). Cross-validation is used to find the optimal similarity measure and the optimal size of the nearest neighbourhood. The final models are then compared to ARM, which computes recommendations by finding rules that imply co-occurrences between items. Something similar is done by Jalili et al. (2018), who compare a number of techniques on

several datasets in order to review evaluation metrics to assess recommendation performance.

This paper adds to the literature by discussing the difference in performance between item-based and user-based recommendations in the case of OCCF problems. Related literature on non OCCF problems suggests that the performances of recommendation techniques are influenced by the evaluation metrics used and the characteristics of the data that is used to compute the recommendations from (Jalili et al., 2018).

The aim of this research is to find the differences in recommendation performance between UBCF, IBCF and ARM, when having only unary purchase data available. The research question can thus be defined as:

> *What is the difference in performance between User-based Collaborative Filtering, Item-based Collaborative Filtering and Association Rule Mining, when computing recommendations for One-Class Collaborative Filtering problems?*

The main results of this paper indicate that IBCF outperforms both UBCF and ARM in terms of performance. UBCF is the least performing method, indicating that item-based similarities might prove more beneficial when there is little to no explicit feedback from users. However, the results show no evidence of a performance gain relative to just always recommending the most popular items when item popularity is highly skewed. This does not seem to be specific to OCCF, since transforming the unary ratings to real ratings also leads to results that do not seem to outperform the model always recommending the most popular items.

The remainder of this thesis is structured as follows: the next section will cover related literature on recommender systems, followed by a detailed description of the data in Section 3. Section 4 describes the methods that are used, along with the resampling technique and evaluation metrics. Section 5 shows the results after which the conclusion of the paper and the discussion on further research can be found in Section 6.

2 Literature Review

Ever since the first research paper on collaborative filtering was written in the mid-1990s, recommender systems have been widely studied (Park et al., 2012). Before diving deeper into the relevant literature on this subject, we first explain some basic theory on the different forms of recommender systems. First, we will cover the concept of user feedback and how this is used in recommender systems. After that, several different models of recommender systems will be explained. To conclude on this section, there will be a summary of the relevant literature for this topic.

2.1 Recommender input

Recommender systems are generally used to compute lists of items that might be of interest to users. These lists are comprised of suggestions that can help users make decisions on which item to buy. As stated by Ricci et al. (2011), the term "items" refers to everything that is being recommended by a recommender system.

The most simple version of a recommender system will offer a ranked list of items, of which the ranking is based on the preferences and constraints of the user (Ricci et al., 2011). The information about these preferences and constraints is obtained through feedback given by users and functions as the input for recommender systems. User feedback can be divided into two categories, namely explicit and implicit feedback (Jannach et al., 2010). Explicit feedback is a very detailed way of obtaining information about the experience a user has had with a certain product. This type of feedback is usually provided by ratings given to or reviews written about items. Recommender systems based on explicit feedback require explicit input from users, which is not always widely available, due to the fact that it takes an extra effort for users to provide it. Implicit feedback on the other hand can be inferred by interpreting the behavior of users and does not require any explicit input from users (Hu et al., 2008). By monitoring the behavior of a user, one can deduce certain conclusions about their preferences. If a user tends to visit a certain page more than other pages, we can assume that this user has an interest in the items on that page. Users buying products without rating or reviewing these items can also be considered implicit feedback. In general, explicit feedback is considered to be the more precise way of collecting information about preferences, but it does have some disadvantages compared to implicit feedback. As stated before, explicit feedback requires explicit input from users, which is not always available. Also, ratings and reviews can become biased when only users with certain emotions regarding the item of interest decide to rate or review that item.

Overall, the information derived from feedback can be divided into three objects: items, users and transactions (Ricci et al., 2011). Transactions can be regarded as the interactions between users and items and they contain the important and useful information about these interactions. The most popular transactions are ratings, which can take on several forms (Schafer et al., 2007). In general, ratings are categorized as scalar ratings, binary ratings or unary ratings. Scalar ratings can either be numerical ratings or ordinal ratings and they provide a scale on which a user can rate items. With binary ratings, users can only choose between two alternatives, positive or negative. Comparing this to unary ratings, this type of rating only indicates whether or not there has been an observable interaction between a user and an item.

After collecting user feedback, the behaviour and preferences can be modeled as a user-item

matrix, where users and items are represented as rows and columns, respectively. These matrices can usually either be a real rating matrix or a binary rating matrix. Real rating matrices have values from a predefined ordinal range for each user-item combination, whereas binary rating matrices value each user-item combination with either a 1 or a 0. In the case of binary rating matrices, binary user feedback leads to a 1 for positive ratings and a 0 for negative ratings. Unary feedback is depicted as a 1 for having purchased an item or a 0 for not having purchased an item.

This thesis uses ratings of the unary sort. The data that is used, originates from an online grocery store and shows the products that were sold for each order. Since there is no information about interests aside from the distinction between products bought and products not bought, the rating matrix that is computed from the user feedback will be a binary rating matrix.

2.2 Recommender systems

There are several methods that compute recommendations, of which the most common ones are Content-Based (CB) filtering and Collaborative Filtering (CF). The main difference between these two methods is that CB filtering is based on the characteristics of items, whereas CF is based on similarities between users or items. CF methods can be further divided into memory- and model-based methods. This section will first cover the CB filtering approach, which will only be explained in short, since it will not be used in the remaining part of this thesis. It is merely explained to give a more complete overview of the possible approaches. The memory- and model-based CF methods will be explained afterwards.

2.2.1 Content-based filtering

When making recommendations, one way is to look at items that were previously used by a user. If a user used an item that has certain characteristics, chances are that he or she will also be interested in another item that has the same characteristics. CB filtering is based on this principle and tries to determine the characterizing features of items. These features can be anything that distinguishes an item from other items. Examples of these features are the genre of a movie, the musician of a song or the topic of a news article. When recommending using CB filtering, the main two pieces of information that are needed are: the characteristics of items and user profiles containing information about the item characteristics of previously used items (Jannach et al., 2010). Let's say we want to recommend a movie to a certain user. We first have to know what kind of movies this user usually watches. The system should hold a user profile on this user, containing information about the characteristics of previously watched movies.

If we assume that this user's profile shows that he or she previously watched comedy movies, then we can recommend movies that are characterized as such. It can be easily deduced that this recommender method mainly relies on the ability to describe the characterizing features of items. We do not necessarily need a much larger user community or a detailed rating history to compute these recommendations.

2.2.2 Memory-based collaborative filtering

The second method is CF, which is based on the idea that users who shared the same preferences in the past, will also share the same preferences in the future (Jannach et al., 2010). If user A has a very similar purchase history compared to user B, then if user A has just bought a product that is not yet seen by user B, we would recommend that product to user B as well.

As stated before, CF can be separated into two main categories, memory- and model-based CF (Breese et al., 1998). Memory-based CF uses similarity measures to find similar users within the full user-item matrix and computes recommendations directly. Model-based CF on the other hand uses the user-item matrix to train models that learn underlying relationships about the data. Recommendations are then based on the predictions of the model and not on the entire user-item matrix. The memory-based models will be explained first.

The most well know memory-based CF methods are the nearest neighbor algorithms (Schafer et al., 2007). These algorithms use the user-item matrix to find similar users through the use of statistical similarity measures. The first algorithm that we will discuss is the user-based nearest neighbor algorithm. This method searches for users that are similar to the user of interest in their behavior or preferences. These similar users are called neighbors and are used to generate predictions about the user of interest. To predict the ratings of user A for a list of items, we use the ratings of the nearest neighbors to compute item scores. This method is based on the assumptions that users will have similar tastes in the future if they had similar tastes in the past and that the preferences of users will be consistent over time (Jannach et al., 2010). One of the challenges for this method lies with data sparsity, which could lead to skewed neighbors that dominate the neighborhood of a user (Schafer et al., 2007). Another challenge is that the similarity measures fail to differentiate between similar ratings that are more meaningful than others. For example, it would be more interesting to know that two users have the same opinion on a more controversial movie than it is to know they both enjoy a movie that almost everyone enjoys. The final challenge is that the calculation of neighborhoods can become computationally less feasible when the amount of users and ratings increases, leading to a scalability problem.

The second algorithm is the item-based nearest neighbor method. This method can be seen

as the transpose of its user-based alternative (Schafer et al., 2007). Item-based nearest neighbor algorithms base their predictions on the similarities between items, instead of similarities between users. These similarities are computed by examining ratings for different items to find items that score similar to the item of interest among the community of users (Sarwar et al., 2001). If we want to predict the rating of an unseen item, we find the already rated items that show similar scoring patterns among the total user pool. The predicted rating for the item of interest will be comparable to the already known ratings of the similar items. One of the benefits of item-based nearest neighbor algorithms is that the total number of correlations used for prediction can be pruned to make computation more feasible and to diminish the effect of skewed correlations dominating predictions (Schafer et al., 2007). However, the downfall of this adjustment is that it leads to less accurate predictions.

2.2.3 Model-based collaborative filtering

The second CF category consists of model-based algorithms. As just discussed, one of the problems for memory-based CF lies with the scalability of the algorithms. Model-based CF tries to tackle this by computing a model that captures the full dataset, but is computationally more feasible when producing recommendations, because of the recommendations now being based on the model instead of the full user-item matrix (Jannach et al., 2010). In many of the model-based CF applications, techniques are used that can be defined as parts of Data Mining processes (Amatriain et al., 2011). Techniques that are very common in model-based CF are dimension reduction techniques (Goldberg et al., 2001; Paterek, 2007), association rule mining (Mobasher et al., 2001; Cho et al., 2002; Lin et al., 2002) and probabilistic classification methods (Miyahara and Pazzani, 2000; Cho et al., 2002; Salakhutdinov et al., 2007).

Next to scalability, another limitation for memory-based CF is data sparsity (Cho et al., 2002). Generally, memory-based CF needs explicit non-binary ratings to produce accurate recommendations. When data is sparse, neighborhoods are harder to compute, leading to less accurate predictions.

Association rule mining can be seen as a solution to sparse data and also tackles the scalability problem. It is a technique that searches for rules that predict the occurrence of items, given the occurrences of other items (Amatriain et al., 2011). Relationships between items are related to the co-occurrence of these items and not causality of any sort. Association rules can be expressed as $X \Rightarrow Y$, with X and Y being sets of items (itemsets). The *support* of $X \Rightarrow Y$ is defined by the percentage of the transactions that contain both X and Y, whereas the *confidence* of that rule is given by how often Y appears in transactions that contain X. By only depending on the

occurrence of items, association rule mining solves the data sparsity problem.

Overall, with association rule mining the aim is to find all rules that satisfy some user-specified minimum values for both the support and the confidence (Cho et al., 2002). The first step in this process is to generate all itemsets that satisfy the minimum support value, also known as *frequent* itemset generation. One of the more common methods for frequent itemset generation is the Apriori algorithm (Agrawal et al., 1994), which is based on the principle that if an itemset is frequent, then all of its subsets must also be frequent. After finding the frequent itemsets, high confidence rules are generated, which are used to produce recommendations, making this method very scalable when the amount of data increases, since it does not rely on the full user-item matrix anymore.

2.3 Literature summary

The difference in performance between user- and item-based CF has been studied extensively in previous literature. An example is given by Sarwar et al. (2001), who analyze several item-based models, each having different similarity measures or recommendation techniques. The results indicate a performance gain in terms of both computation time and accuracy when compared to user-based models. In addition, Deshpande and Karypis (2004) present an item-based approach to collaborative filtering, leading to qualitatively comparable recommendations that are significantly faster to compute than the user-based approach. Furthermore, Jalili et al. (2018) compare several collaborative filtering approaches by testing them on multiple datasets. They use different evaluation metrics and conclude that performance highly depends on the characteristics of the dataset and which evaluation metric is used.

Most of the previous literature, however, assume a richer set of user feedback available to compute recommendations from. In the case of real ratings, a lot of studies use datasets containing ratings on an ordinal scale, whereas the papers with binary ratings assume positive preferences for ratings of 1 and negative preferences for ratings of 0. In the case of OCCF, however, the used dataset usually contains less to no information about the negative preferences of users, as a rating of 0 indicates the absence of preference for that item. Previous literature on OCCF problems touches on several subjects, such as handling unlabeled instances or solving the class imbalance problem (Pan et al., 2008). To our knowledge however, the difference between user- and item-based nearest neighbour algorithms in the OCCF setting has not yet been covered.

3 Data

This section introduces the data that is used for this research. The Instacart Online Grocery Shopping dataset provides purchase data from an online grocery store and is publicly available online (Instacart, 2017). The data is used to investigate the possibilities of increasing cross-selling opportunities by implementing recommender systems. The dataset contains information about the content of each basket, the time of placement and how long it has been since the previous order for a given customer. The product assortment is comparable to an average grocery store and regarding the customers, there is no information available to set up demographic profiles.

3.1 Data description

The dataset contains information about 3,421,083 orders from 206,209 different customers. Due to the size of the dataset and regarding the computational feasibility, we use a sample of 10% from this dataset. The sample is drawn by randomly selecting 20,620 customers, leading to a total number of 331,451 orders. For each order, we know which products were added to the basket, where products can be added multiple times to a single basket, leading to the fact that the number of items will always be equal or greater than the number of different products for each order. Among these 331,451 orders, a total number of 3,336,718 products were sold. On average, customers order (3,336,718/331,451 $\approx$) 10.07 products per order.

The number of different products is 49,685, which is why this paper does not consider unique products in computing recommendations. Using unique products when computing the user-item matrix would lead to an infeasible large matrix, where data would be sparse and similarities would be hard to compute. Therefore, this thesis uses sub-classes of items, which can be described as groups of similar products. The sub-classes are based on two different hypernyms, where each product is categorized by the aisle and department it belongs to. Table 1 gives an overview of the number of different sub-classes per hypernym.

Table 1: Number of sub-classes per hypernym

	Department group	Aisle group
# of sub-classes	21	134

By using sub-classes of products instead of unique products in determining unique items, we decrease the overall number of different items per order. If two different products belong to the same sub-class, they will be counted as one item. Looking at the resulting user-item matrix, we will have a matrix where the rows represent the users and the sub-classes of products will be

the columns. If a customer orders an item that belongs to a certain sub-class, there will be a 1 in the matrix for that sub-class and a 0 otherwise.

Regarding the different hypernyms, it is important to decide which sub-classes will be used for making recommendations. Improving cross-selling may lead to an improvement of the durable income for a retailer, so making recommendations within the right sub-classes of products is essential. The different sub-classes within the department groups seem to be quite general, which is only logical since there are only 21 sub-classes for 49,685 different products. Comparing this to the aisle group, the sub-classes in this group still need to capture a large amount of products within 134 sub-classes, but they seem much more precise than their department counterparts. Using the aisle groups will lead to more precise recommendations than using the department group, which is why this paper will focus only on the aisle groups.

When looking at the aisle groups, it is obvious that the number of orders remain unchanged. Furthermore, the number of aisle groups ordered is 134, meaning that all aisle groups are being ordered. Next to this, it is viable to know how many different aisle groups are being ordered by individual customers. Figure 1 shows the distribution between customers of the number of different aisle groups.

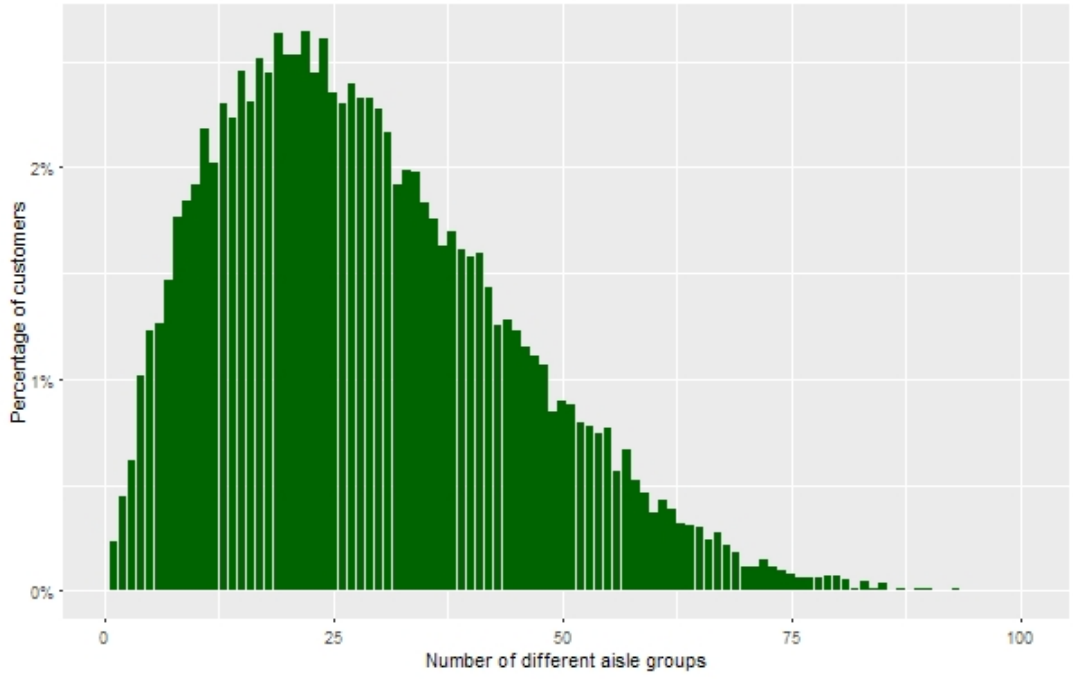

Figure 1: Percentage of customers vs. number of different aisle groups

It is observable that most customers order products from around 25 different aisle groups and that only a few order products from more than 75 different aisles. Figure 1 also shows that there is room for cross-selling opportunities, since buying products from more than 25 aisle groups is

already being done by a lot of customers. Figure 2 strengthens this idea by showing that nearly half of all customers order products from only 1 to 25 different aisle groups.

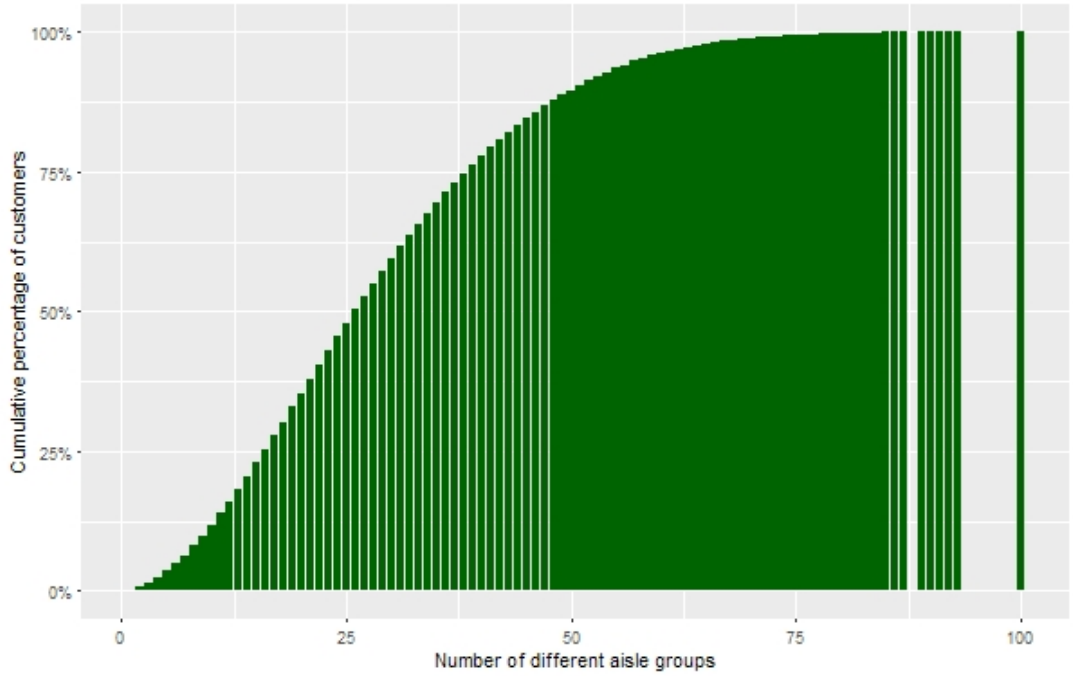

Figure 2: Cumulative percentage of customers vs. number of different aisle groups

Another thing made clear by looking at Figures 1 and 2 is that most customers will have more than one aisle group from which they buy products, leading to a dense user-item matrix. This will make for a relatively easy computation of similarities between customers' purchase histories.

Table 2 shows the ten aisle groups that were purchased the most. The third column shows the size of each aisle group in proportion to the total number of purchases. It shows that the most popular items make up for a very large portion of the total purchases and that the purchases are not distributed equally among the different aisle groups.

Table 2: Most popular aisle groups

Rank	Aisle group	Purchase percentage (%)
1	Fresh fruits	11.13
2	Fresh vegetables	10.56
3	Packaged vegetable fruits	5.46
4	Yoghurt	4.50
5	Packaged cheese	3.05
6	Milk	2.75
7	Water seltzer sparkling water	2.65
8	Chips pretzels	2.23
9	Soy lactosefree	1.99
10	Bread	1.78

3.2 Train and test set

In order to test the performance of each model, the data is split into a train set and test set. Splitting the data is done by taking a random 60/40 split, where the train set consists of 60 percent of the total users, randomly selected from the pool of unique users. The characteristics of both subsets are shown in Table 3. The train set consists of 12372 users and 1,996,160 total purchases, leading to an average number of purchases per user of 161.35. The test set contains 8248 users and 1,340,558 total purchases, leading to an average of 162.53 purchases per user. Table 3 shows that all aisle groups appear in both subsets and that the average number of purchases per user is nearly the same for both subsets.

Table 3: Characteristics of train and test set

Subset	Users	Purchases	Average purchases per user	Aisle groups
Train data	12372	1996160	161.35	134
Test data	8248	1340558	162.53	134

Both the train and test sets are used to compute user-item matrices that model the users' purchase behaviour. The users are represented in the rows of these matrices, while each column will depict one of the aisle groups. If a user has purchased a certain item in the train set, the corresponding user-item combination will be filled with a value of 1. All other user-item combinations in the matrix will receive the value of 0.

When evaluating the results of this paper, several alternative subsets of the data are sampled to compare the main results with. The characteristics of these subsets are not discussed here,

since they follow a similar pattern as the main dataset. They are also not in the main scope of this paper, but merely function as comparison.

In addition, in order to test whether our results are OCCF specific, we compute recommendations on a dataset where the unary ratings are transformed to real ratings as well. This transformed dataset originates from the main dataset of this paper, so it is expected to have similar characteristics. Therefore, the transformed dataset is not explained in detail here as well.

4 Methodology

This section describes the different methods that are used to generate recommendations. We start with the memory-based CF approaches, User-Based CF (UBCF) and Item-Based CF (IBCF), after which we'll discuss the model-based approach of Association Rule Mining (ARM). The memory-based CF approaches use similarity measures to find a user-specified number of similar users. In order to find the best similarity measure and the optimal number of similar users, we will use cross-validation (CV). In addition, CV is also used to estimate the optimal number of items to recommend. The optimal values for these parameters are computed by comparing the performance of the corresponding methods, which is measured by several evaluation metrics.

First, Section 4.1 explains CV and its application in this paper. After this, the memory-based approaches and different similarity measures are explained in Section 4.2. Subsequently, the ARM approach is discussed in Section 4.3, after which we conclude with describing the evaluation metrics in Section 4.4.

Throughout this thesis, we will use the same notation for all methods. Formally, there are $\mathcal{U}$ different users and $\mathcal{I}$ different items, where $\mathcal{U} = \{u_1, u_2, ..., u_m\}$ and $\mathcal{I} = \{i_1, i_2, ..., i_n\}$. The corresponding ratings are captured in an $m \times n$ rating matrix $\mathbf{R}$ in the case of UBCF and an $n \times m$ rating matrix $\mathbf{R}^T$ in the case of IBCF. We define $\mathbf{R} = (r_{jk})$ as

$$r_{jk} = \begin{cases} 1, & \text{if user } u_j \text{ already purchased item } i_k \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

4.1 Resampling methods

Resampling methods are used to solidify the confidence of performance measures. By repeatedly drawing samples from a data set, we are able to fit and evaluate each model multiple times. This way, we are able to tune the parameters used in the models and make a more confident

estimate of the performances. The resampling method used in this paper is cross-validation (CV), where each model is evaluated a number of times, after which the results are averaged. The parameters that are tuned specifically for the memory-based models are the similarity measures and the number of nearest neighbours. In addition, we also tune the number of items recommended for all models.

With k-fold CV, the data set is split into k subsets, all being approximately the same size (Kohavi et al., 1995). Each time, one of the subsets is used as validation set, while the remaining k - 1 subsets are used to train the model on. After repeating this process k times, the results are averaged, producing more robust results. The error measure that is used is the mean absolute error (MAE), which can be defined as

$$MAE = \frac{1}{N} \sum_{i=1}^{N} |\epsilon_i|, \tag{2}$$

where N is the total number of items available for recommendation and ϵ_i is the absolute error for item i. This absolute error is computed by taking the difference between the predicted value of each item and its actual value, which will always be either 0 or 1.

4.2 Nearest neighbour recommendations

The main idea with memory-based collaborative filtering is that similar users have similar preferences (Ning et al., 2015). We expect users who show similar purchase histories to also prefer the same items in the future. The most common memory-based collaborative filtering approach is the nearest neighbour approach, which can be divided into user-based nearest neighbours and item-based nearest neighbours. Both methods use similarity measures to find similar users, or nearest neighbours. This section will first explain the user-based approach, along with the different similarity measures that are being considered, after which we will cover the item-based alternative.

4.2.1 User-based nearest neighbour recommendations

The first memory-based approach considered is the user-based nearest neighbour recommendation method. In general, this method predicts the rating r_{jk} of user u_j for item i_k, based on the ratings of users that are similar to user u_j for the same item i_k (Ning et al., 2015). These similar users, also known as nearest neighbours, are the users that show the most similarity in purchase behaviour.

To find the neighbourhood of similar users, we use similarity measures and select a specified number of nearest neighbours, also known as the k-nearest-neighbours (kNN) algorithm. Two

common similarity measures are the Pearson correlation coefficient and the Cosine similarity, which are defined by

$$Pearson(\mathbf{x}, \mathbf{y}) = \frac{\text{cov}(\mathbf{x}, \mathbf{y})}{\sigma_{\mathbf{x}}\sigma_{\mathbf{y}}} \tag{3}$$

and

$$Cosine(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{||\mathbf{x}|| * ||\mathbf{y}||}, \tag{4}$$

where $\mathbf{x} = r_x$ and $\mathbf{y} = r_y$ describe the purchase behavior of users u_x and u_y through their row vectors in $\mathbf{R}$. The covariance between $\mathbf{x}$ and $\mathbf{y}$ is given by cov($\mathbf{x}$,$\mathbf{y}$), while $\sigma_{\mathbf{x}}$ refers to the standard deviation of $\mathbf{x}$. The $\cdot$ implies the vector dot product and $||\mathbf{x}||$ is the Euclidian length of $\mathbf{x}$.

In addition to the Pearson correlation coefficient and the Cosine similarity, we also consider the Jaccard index for measuring similarity. When calculating similarities between two users with (3) or (4), we only use the items that are rated by both users. For binary data however, this would not lead to a useful measure, since rated items can only have a value of one. This can be solved by assuming that users dislike the items that they do not buy. This does, however, tend to lead to significant errors for new users with very few ones in their purchase history vectors. To deal with this, the Jaccard index only focuses on ones, thereby preventing the problem with having too many zeroes. This similarity measure is given by

$$Jaccard(\mathcal{X}, \mathcal{Y}) = \frac{|\mathcal{X} \cap \mathcal{Y}|}{|\mathcal{X} \cup \mathcal{Y}|}, \tag{5}$$

where $\mathcal{X}$ and $\mathcal{Y}$ indicate the sets of items that have a value of one for users u_x and u_y respectively. The similarity between users is measured by computing the intersection between $\mathcal{X}$ and $\mathcal{Y}$, which is subsequently divided by the union of both users' purchase history.

After computing the similarities between users, we define the neighbourhood $\mathcal{H} \subset \mathcal{U}$ by taking the k nearest neighbours. The users within this neighbourhood are then used to predict the ratings for the active user by taking the weighted average of their ratings. The predicted ratings are depicted as non-normalized scores through

$$s_{jk,UBCF} = \sum_{l \in \mathcal{H}} w_{jl} r_{lk}, \tag{6}$$

where s_{jk} is the predicted score for item i_k by user u_j, r_{lk} is the rating for item i_k by user u_l and w_{jl} is the similarity between users u_j and u_l, and functions as a weight. After computing the scores for each item i and user u, we generate a top-N list, containing the N highest scoring items for each user.

4.2.2 Item-based nearest neighbour recommendations

Contrary to the user-based CF approach, item-based CF does not use similarities between different users to predict ratings. Instead, item-based nearest neighbour recommendations originate from the similarities between items, which are inferred from the ratings matrix $\mathbf{R}$. These similarities are computed using the same similarity measures as with UBCF. The only difference here is that we use a $\mathcal{I} \times \mathcal{U}$ ratings matrix as input for the similarity matrix.

The predicted IBCF scores are calculated by totaling the similarities with the items that were already purchased by the user of interest for each item available for recommendation. The scores are defined by

$$s_{jk,IBCF} = \sum_{i \in \mathcal{I}} w_{ik} r_{ji}, \tag{7}$$

where r_{ji} is the rating for item i_i by user u_j and w_{ik} is the similarity between items i_i and i_k. As with UBCF, we limit the size of the neighbourhood by specifying a value beforehand. This value determines the number of items that are considered to be an item's nearest neighbours. All other similarities are discarded and are not used in computing the score. The remaining similarities are then used to compute a top-N list, similarly to UBCF.

4.2.3 Nearest neighbour parameter tuning

In order to find the best performing nearest neighbour algorithm parameters, we use CV to see which combination of similarity measure and number of nearest neighbours results in the best performance. For both methods, we compare the Pearson correlation coefficient, the Cosine similarity and the Jaccard measure. Regarding the optimal number of nearest neighbours, we consider only three levels and compare those. The first level is considered the rule of thumb for choosing k and is found by taking the square root of the number of users (Hassanat et al., 2014). The two other values are values that are relatively close to the value from the rule of thumb.

After tuning the similarity measure and the number of nearest neighbours, we compare the best performing UBCF and IBCF models to the ARM model in terms of performance and accuracy.

4.3 Association rule mining

Association rule mining (ARM) is a technique that tries to find rule-like relationships between the occurrences of items (Jannach et al., 2010). Each user's purchase history is seen as a transaction, consisting of all items in $\mathcal{I}$ with a value of 1 for that user specifically. More formally, transaction j can be defined as $\mathcal{T}_j = \{i_k \in \mathcal{I} | r_{jk=1}\}$ and the full transaction dataset is defined

as $\mathcal{D} = \{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_U\}$, where the number of different users is given by U. The goal with ARM is to find relationships between different items, that can be used as rules. These rules are often written as $\mathcal{X} \rightarrow \mathcal{Y}$, where both $\mathcal{X}$ and $\mathcal{Y}$ are subsets of $\mathcal{I}$ and $\mathcal{X} \cap \mathcal{Y} = \emptyset$. The definition of such a rule would be that whenever the items of subset $\mathcal{X}$ occur in a transaction, we also expect the items of subset $\mathcal{Y}$ to be in that same transaction.

In order to convert the found rules into a top-N recommendations list, we use two measures that determine the significance of each rule. We use these measures to focus on the more important rules and exclude insignificant ones. The first measure is the *support* of a rule, which gives an indication of the probability of two itemsets co-occurring in a transaction and is given by

$$support(\mathcal{X} \rightarrow \mathcal{Y}) = \frac{\text{number of transactions containing } \mathcal{X} \cup \mathcal{Y}}{\text{total number of transactions in } \mathcal{D}}. \tag{8}$$

The second measure is the *confidence* of a rule, which measures the probability of itemset $\mathcal{Y}$ occurring in a transaction, given the fact that itemset $\mathcal{X}$ is also in that transaction. The confidence is given by

$$confidence(\mathcal{X} \rightarrow \mathcal{Y}) = \frac{\text{number of transactions containing } \mathcal{X} \cup \mathcal{Y}}{\text{number of transactions containing } \mathcal{X}}. \tag{9}$$

After finding the rules that appear to be significant, we select the set of rules of which the user of interest has purchased all items from $\mathcal{X}$. Subsequently, the union of items appearing in $\mathcal{Y}$ of the associated rules is computed, of which the already purchased items are excluded. The top-N list is created by sorting the resulting union of items by their confidence measure and selecting the top N items.

4.4 Evaluation metrics

In order to evaluate the performance of our recommender systems, we compare the computed top-N-lists with the actual purchased items in our test set. The scores that are computed using (6) and (7) are not binary, so after computing the top-N lists, we consider all items that appear in the list to have a predicted rating of 1. This is done since the actual purchases in the test set are binary, depending on whether they are purchased or not. After assuming only binary values, the final UBCF and IBCF models are compared to the ARM model in terms of performance. The first measure is the MAE, using (2), which measures the predictive accuracy of the models by computing the differences between the predicted ratings and the actual purchases.

Next to accuracy, another common technique in evaluating recommender systems is through the use of a confusion matrix, shown in table 4. The values in the confusion matrix correspond to the four different possible classification outcomes when comparing actual to predicted values.

TP corresponds to the items that are recommended in the top-N lists and purchased in the test set as well. FP indicates the items that are recommended, but not purchased in the test set. FN shows the items that were not in the top-N lists, but did get purchased in the test set and TN corresponds to the items that were not recommended and not purchased as well.

Table 4: Confusion matrix

		Actual	
		Positive (1)	Negative (0)
Predicted	Positive (1)	True Positive (TP)	False Positive (FP)
	Negative (0)	False Negative (FN)	True Negative (TN)

From Table 4 several different measures can be derived to evaluate the performance of the recommender systems. The first measure considered is the *precision* of a model, given by

$$precision = \frac{\text{correctly recommended items}}{\text{total recommended items}} = \frac{TP}{TP + FP}, \tag{10}$$

where $TP + FP = N$. Precision measures the percentage of the total number of recommendations that is actually purchased as well. Evaluating recommender systems on their precision is important, because this measure gives an indication of the conversion rate of the recommendations made. Knowing the effectiveness of a model makes it easier to decide which model to use in a given situation.

The second measure is the *recall* of a model, which indicates how many of the actually purchased items were recommended as well. The recall of a model is given by

$$recall = \frac{\text{correctly recommended items}}{\text{total useful recommendations}} = \frac{TP}{TP + FN}, \tag{11}$$

where $TP + FN$ corresponds to the total number of items purchased in the test set.

Both these measures are considered valuable in evaluating recommender systems, but their conflicting nature poses a challenge as to finding the optimal trade-off value between both. A popular way of doing that is by using the *F-measure* (Jalili et al., 2018). This measure is considered the harmonic mean between precision and recall and is given by

$$F\text{-}measure = \frac{2}{1/Precision + 1/Recall}. \tag{12}$$

In addition to the described measures, another way of comparing different classification methods is by using the *Receiver Operating Characteristic* (ROC) (Konstan et al., 1997). The ROC is shown as a plot with each model's *true positive rate* (TPR) and *false positive rate* (FPR) on the y-axis and x-axis, respectively. The TPR of a model is its probability of detection and is equivalent to the model's recall. The FPR of a model corresponds to the probability of false

alarm, which is computed by $\frac{FP}{FP+TN}$. In general, models can be compared by looking at their areas under the ROC-curve. The method with the largest area is commonly considered the method that has the best performance.

5 Results

This section will cover the results of the paper. Using CV, the optimal number of nearest neighbours (kNN) and best performing similarity measure are found for UBCF and IBCF. The best performing models, together with the ARM model and two benchmark models, are then compared based on the chosen evaluation metrics when applied to the test data set.

5.1 Nearest neighbour CV results

The results of the CV for the UBCF and IBCF models are shown in Table 5. The two nearest neighbours methods are used to compute top-10 recommendation lists while using different similarity measures and values of k. The resulting MAE for each model, computed using (2), is used to determine the optimal value for k and find the best similarity measure. First off, Table 5 shows that, for the UBCF models, there is no difference in MAE between the used similarity measures, suggesting that all similarity measures choose the same neighbourhood of nearest neighbours. It is also observable from Table 5 that increasing the number of nearest neighbours does not lead to a lower MAE. This seems counter-intuitive, since increasing the size of the nearest neighbourhood is expected to result in a lower error rate, due to the model having more similar users to compute predictions. A possible explanation for this could be that the predicted ratings are mostly based on a smaller number of nearest neighbours. In that case, these neighbours have a relatively strong weight in the prediction, whereas the users that have lower similarities contribute significantly less in the prediction. Adding more users that only have a little influence does not change the outcome of the total prediction. An alternative explanation would be that the optimal number of k lies far outside the chosen range of values for k.

Looking at the absolute values of the IBCF models, we notice a slight decrease in MAE when increasing $k = 50$ to $k = 111$ and a slight increase when increasing further to $k = 200$. The differences between these levels are so small, however, that we can not assume a statistically significant difference in performance when increasing the size of the nearest neighbourhood within this range of k. Regarding the similarity measures, there does seem to be a performance difference between the similarity measures, where Pearson's correlation coefficient is outperformed by the Cosine and Jaccard measure. The two last-mentioned measures show only a very small

difference when $k = 200$, but the size of the difference leads to the assumption that there is no statistically significant difference in performance between both.

Table 5: MAE of UBCF and IBCF models in CV for $N = 10$

	UBCF			IBCF		
	Pearson	Cosine	Jaccard	Pearson	Cosine	Jaccard
$k = 50$	0.190	0.190	0.190	0.199	0.179	0.179
$k = 111$	0.190	0.190	0.190	0.196	0.178	0.178
$k = 200$	0.190	0.190	0.190	0.198	0.178	0.179

In order to get a better image of the difference in performance between all models, Table 6 compares all models on their precision, recall and a combination of both through the F-measure. Table 6 shows that, regarding the UBCF models, there is no difference in performance between the different similarity measures. The only difference observable is so small, that we assume no significant difference in performance. In addition, increasing the number of nearest neighbours also does not lead to an increase in any of the performance measures. Looking at the IBCF models, Pearson's correlation coefficient is again outperformed by the Cosine and Jaccard measure for all levels of k. However, comparing the Cosine and Jaccard measure for IBCF in Table 6 implies assuming no statistically significant difference between both.

Table 6: Precision, recall and F-measure of UBCF and IBCF models in CV for $N = 10$

		UBCF			IBCF		
		Pearson	Cosine	Jaccard	Pearson	Cosine	Jaccard
$k = 50$	Precision	0.500	0.500	0.501	0.440	0.570	0.572
	Recall	0.204	0.204	0.204	0.180	0.233	0.234
	F-measure	0.290	0.290	0.290	0.255	0.331	0.332
$k = 111$	Precision	0.500	0.500	0.500	0.459	0.578	0.576
	Recall	0.204	0.204	0.204	0.187	0.236	0.235
	F-measure	0.290	0.290	0.290	0.266	0.335	0.334
$k = 200$	Precision	0.500	0.500	0.500	0.447	0.574	0.572
	Recall	0.204	0.204	0.204	0.183	0.234	0.234
	F-measure	0.290	0.290	0.290	0.259	0.333	0.332

Based on Tables 5 and 6, the best performing model for each method is chosen. For the UBCF model, we pick the one with $k = 50$ in combination with the Jaccard similarity measure. Since we cannot assume any statistically significant differences in performance ,the choice is merely

based on the fact that UBCF becomes less computationally feasible when increasing the number of nearest neighbours. The Jaccard measure is chosen, because it is specifically designed for binary data. Regarding the IBCF model, we choose the combination of $k = 111$ and the Cosine measure. Setting k to a value of 111 corresponds to the rule of thumb, given by Hassanat et al. (2014), and compared to UBCF, IBCF tends to be a lot faster to compute, so increasing the size of the nearest neighbourhood has little impact on the computation time. In addition, the choice between the Cosine and Jaccard measure seems arbitrary, so we simply choose the highest F-measure for $k = 111$ in Table 6.

5.2 Test set results

Table 7 shows the results of the different models when performed on the test data. Each model computes a top-10 recommendation list, after which the items on these lists are compared to the actual purchases in the test set. In order to assess the usefulness of each model, two benchmark methods of making recommendations are displayed as well. The first benchmark method is to recommend items randomly and the second method is to always recommend the most popular items. Recommending items randomly is generally not considered to be a great idea, which is also shown in Table 7. It has the highest MAE and lowest precision, recall and F-measure, which comes at no surprise. Always recommending the most popular items, on the other hand, does seem to yield relatively accurate recommendations. The values of the performance measures for this method are second to best, indicating that a large portion of the users does tend to purchase at least some number of items from the 10 most popular items list.

Regarding the nearest neighbour methods, it is observable in Table 7 that the IBCF model outperforms the UBCF model. The results show that the IBCF model has the highest precision, recall and F-measure, and the lowest MAE, thereby outperforming all other models. The UBCF model on the other hand, shows results that are only slightly better than those of recommending items randomly. It is surprising to see our UBCF model performing relatively poorly, as literature suggests that user-based recommendations tend to outperform most model-based approaches in terms of accuracy. A possible explanation could be that users' preferences in terms of grocery shopping are hard to predict when based on similar purchase behaviour, due to the lack of strong patterns in behaviour. This idea is strengthened when assessing the user-based similarity matrix, where most similarities between users are relatively low. This indicates that most users follow a rather unique purchase pattern, making it hard to predict their future purchases purely based on purchase history. Comparing this to the IBCF results, it seems that similarities between items are a better way of predicting future purchases.

Finally, the results of the ARM model are in between the UBCF and IBCF model. This model seems to outperform the UBCF model, but performs worse than the IBCF model. The performance of ARM relative to UBCF strengthens the idea that relationships between items are more suitable for making recommendations than the relationships between users. In addition, the performance difference between ARM and IBCF shows that there is no real performance gain in finding association rules over item to item similarities, while ARM does take significantly longer to compute.

Table 7: Results on test data for UBCF, IBCF, ARM and benchmark models with $N = 10$

Model	Similarity measure	kNN	MAE	Precision	Recall	F-measure
Random items			0.240	0.188	0.076	0.108
Popular items			0.179	0.578	0.234	0.333
User-based CF	Jaccard	50	0.208	0.396	0.160	0.228
Item-based CF	Cosine	111	0.179	0.580	0.235	0.334
Association rules			0.186	0.537	0.217	0.309

Until now, we examined the performance of the different models when $N = 10$. Figure 3 shows the ROC-curve of all models for $N = (1, 3, 5, 10, 15, 20)$, highlighting the difference in performance between UBCF, IBCF and ARM. It is observable that when $N = (1, 3, 5)$, UBCF performs only slightly better than the benchmark model recommending items randomly. After the $N = 5$ mark, the UBCF performance increases, but it never reaches the level of IBCF. In comparison, looking at IBCF and ARM, the strongest performance increase is when N increases from 1 to 3 and from 3 to 5. Increasing N any further does not seem to lead to better recommendations. Figure 3 also shows that our popular items benchmark model performs almost identical to the IBCF model. This does not, however, indicate that we are better off always just recommending the most popular items. It is very probable that over a given time period, almost every user will purchase one or more of the most popular items, regardless of whether these items are recommended or not. This is also reflected in Table 2, which shows the most popular items and their purchase percentage. Taking the top 10 most popular items leads to a total purchase percentage of 46.08%, indicating that almost half of all purchases belong to the most popular items. The results for always recommending the most popular items are therefore no surprise. In addition, one of the major downfalls of recommending only popular items is that this does not stimulate cross-selling.

As stated above, there is a strong skewness in the number of purchases per aisle group, which is a probable cause of the relatively strong performance of always recommending the ten

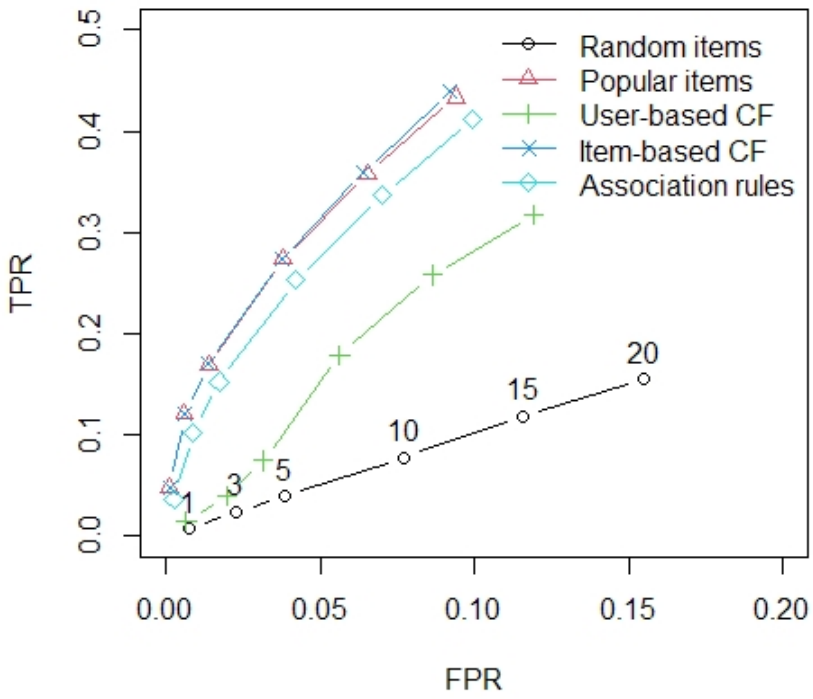

Figure 3: ROC-curve of UBCF, IBCF, ARM and benchmark models for different values of N

most popular items. Table 8 shows the results on the test set when combining UBCF, IBCF and AR with item popularity. Notice that the performance becomes almost identical for all models, indicating that these popular items have a strong influence on the recommendations. For the UBCF and AR model, adding item popularity leads to an increase in performance, but not beyond the model that just recommends the most popular items. Adding item popularity does not seem to have an impact on the IBCF model, which shows the same performance metric values as without.

Table 8: Results on test data when adding item popularity, with $N = 10$

Model	Similarity measure	kNN	MAE	Precision	Recall	F-measure
Popular items			0.179	0.578	0.234	0.333
UBCF-popular	Jaccard	50	0.178	0.575	0.235	0.334
IBCF-popular	Cosine	111	0.177	0.580	0.237	0.336
AR-popular			0.177	0.578	0.236	0.336

Another way to gauge the impact of item popularity is by computing recommendations on a subset of the data that excludes the most popular items. Table 9 shows the results of the different models when performed on a subset of the test data, excluding the top 50 percent of all aisle groups, based on popularity. The first thing to notice is that, in general, excluding these most

popular items leads to a significant decrease in performance. Especially the UBCF method seems to suffer, now becoming less effective than just recommending random items. Furthermore, the precision of each model experiences a substantive decrease, while the values for recall seem to increase. This increase, however, can be expected when the total number of items available for recommendation is decreased. Comparing Tables 7 and 9, based on the F-measure, almost all models suffer a decrease in performance, with the random items model being the exception. However, Table 9 does indicate that the popular items model suffers relatively more than the IBCF and AR models. Both now seem to outperform the benchmark model, affirming the thought that the skewness in the aisle groups affects the recommendation performance.

Table 9: Results on subset when excluding most popular items, with $N = 10$

Model	Similarity measure	kNN	MAE	Precision	Recall	F-measure
Random items			0.228	0.075	0.186	0.107
Popular items			0.207	0.134	0.331	0.190
User-based CF	Jaccard	50	0.240	0.043	0.107	0.062
Item-based CF	Cosine	111	0.195	0.166	0.412	0.237
Association rules			0.204	0.141	0.348	0.200

5.3 Alternative subset results

In order to see if our results are specific for this dataset only, we also compute recommendations on subsets of the total dataset. By taking subsets, we aim to find data that is less skewed by item popularity. The subsets are sampled from the original dataset, based on when the orders were placed. Table 10 gives an overview of the recommendation results for each of the different subsets.

Table 10: Results on alternative subsets, with $N = 10$

Subset	Model	MAE	Precision	Recall	F-measure
Weekend	User-based CF	0.153	0.320	0.198	0.245
	Item-based CF	0.132	0.457	0.283	0.349
	Association rules	0.139	0.415	0.256	0.317
Morning	User-based CF	0.152	0.308	0.195	0.239
	Item-based CF	0.132	0.441	0.279	0.342
	Association rules	0.138	0.400	0.253	0.310
Afternoon	User-based CF	0.159	0.380	0.210	0.271
	Item-based CF	0.142	0.185	0.269	0.346
	Association rules	0.149	0.442	0.245	0.315
Evening	User-based CF	0.139	0.242	0.190	0.213
	Item-based CF	0.116	0.390	0.306	0.343
	Association rules	0.121	0.355	0.278	0.312
Night	User-based CF	0.097	0.235	0.337	0.277
	Item-based CF	0.098	0.233	0.335	0.274
	Association rules	0.102	0.208	0.299	0.245

It is clear from Table 10 that the results are very similar for most of the subsets. Except for the orders placed at night, each subset shows that IBCF outperforms UBCF and AR, where UBCF performs worse than AR. Looking at the F-measure, IBCF and AR seem to perform similarly for each subset, while the performance of UBCF fluctuates between the subsets. Regarding the orders placed at night, IBCF and AR seem to perform significantly worse than with the other subsets, while UBCF shows the best performance when compared to the other subsets. Looking at Table 11, it is no surprise that the recommender systems based on the subsets show similar performances. Table 11 shows clearly that all subsets follow an almost identical distribution of the most popular items, only differing in size. This is unfortunate, since it does not tell a lot about the level of influence of the skewed item popularity. The only thing that can be said is that it seems like IBCF and AR suffer more from datasets that have significantly fewer datapoints than UBCF.

Table 11: Distribution of the ten most popular items for each subset

	Weekend		Morning		Afternoon		Evening		Night	
Rank	n	%	n	%	n	%	n	%	n	%
1	135,747	11.6	112,923	11.4	172,840	11.1	49,391	11.3	1,066	11.6
2	131,544	11.3	101,866	10.3	167,134	10.7	46,721	10.7	1,009	11.0
3	67,368	5.8	53,681	5.4	85,279	5.5	24,451	5.6	504	5.5
4	52,479	4.5	45,401	4.6	65,587	4.2	19,433	4.5	415	4.5
5	35,482	3.0	29,232	3.0	47,757	3.1	12,802	2.9	265	2.9
6	31,423	2.7	28,227	2.9	41,080	2.6	11,804	2.7	234	2.6
7	28,806	2.5	26,734	2.7	39,826	2.6	10,684	2.5	215	2.3
8	23,964	2.1	21,787	2.2	34,983	2.2	8,706	2.0	204	2.2
9	22,920	2.0	19,697	2.0	29,662	1.9	8,672	2.0	182	2.0
10	21,533	1.8	17,972	1.8	28,610	1.8	8,165	1.9	176	1.9
Total	551,266	47.2	457,520	46.2	712,758	45.7	200,829	46.1	4,270	46.5

5.4 Transformed rating results

The results thus far show no evidence of a performance gain relative to always recommending the most popular items. To see whether this is specific to OCCF datasets only, recommendations are also computed on a dataset where the unary ratings are transformed to real ratings. This transformation is done by constructing the user-item matrix in a way that tries to capture the relative item preference through the quantity of the purchases. Each user-item combination in the matrix shows the number of times an item was purchased by a specific user, after which these values are normalized to a scale between 1 and 5. By transforming the unary ratings to real ratings, we are able to change the recommendation setting where we are no longer dealing with OCCF. Table 12 compares the results of computing a top-10 list on these real ratings when using UBCF, IBCF and the benchmark models. The similarity measures and values of kNN are derived through CV and the association rules model is not included, as this model is not suited for real ratings since it only uses the occurrences of items instead of ratings.

Table 12: Results on transformed ratings dataset, with $N = 10$

Model	Similarity measure	kNN	MAE	Precision	Recall	F-measure
Random items			0.238	0.192	0.078	0.111
Popular items			0.189	0.507	0.207	0.294
User-based CF	Cosine	282	0.216	0.330	0.134	0.191
Item-based CF	Pearson	142	0.204	0.398	0.138	0.205

In general, Table 12 shows that all models perform worse than before transforming the ratings, as depicted in Table 7, indicating that adding the relative preferences of users between items does not always necessarily lead to an increase in recommendation performance. Furthermore, again the best performing model in this setting is the model recommending the most popular items overall, suggesting that our previous results are not OCCF specific. The difference between UBCF and IBCF, however, does seem to be smaller in this setting than before transforming the ratings. A possible explanation of this could be that UBCF is more effective at using relative item preferences of users.

6 Conclusion

The main goal of this paper was to evaluate the difference in performance between user-based and item-based recommendations when applied to One Class Collaborative Filtering problems. A dataset from an online grocery shopping market was used to find an answer to the following main research question:

> *What is the difference in performance between User-based Collaborative Filtering, Item-based Collaborative Filtering and Association Rule Mining, when computing recommendations for One-Class Collaborative Filtering problems?*

To answer this research question, several methods were used to compute top-N lists, recommending items based on previous purchase behaviour. The included methods were User-based Collaborative Filtering, Item-based Collaborative Filtering and Association Rules, of which the performances were also compared to those of two benchmark models, the random items and the popular items models. The initial results showed that IBCF and AR outperformed the UBCF model, where IBCF had the best performance. The popular items benchmark model, however, outperformed all other models. A possible explanation could be found in the distribution of the item popularity, which was highly skewed. This effect was tested by computing recommendations through hybrid models, that included item popularity, and by computing recommendations

on a subset excluding the most popular items. The hybrid models all showed similar performance to the popular items benchmark model and the results on the subset excluding the most popular items showed a relatively smaller decrease in performance for IBCF and AR than for UBCF and the popular items benchmark model, implying an effect of the distribution of item popularity on recommendation performance. To test this further, several alternative subsets were taken from the original dataset, based on the day and time of the day orders were placed. However, these alternative subsets all showed similar item popularity distributions, therefore not really contributing to testing the effect of item popularity distribution. Finally, in order to test whether these results are OCCF specific, recommendations were computed on a dataset where the unary ratings were transformed to real ratings by using the quantity of the purchases. The results on the real ratings dataset showed no statistically significant difference in performance between UBCF and IBCF, but the popular items model outperformed all other models in the non OCCF setting as well.

Answering the main research question can thus be done by summarizing the above. Based on the dataset used, item-based recommendations seem to yield better performance than user-based recommendations, where AR is outperformed by IBCF. This conclusion, however, is subject to the influence of skewed item popularity.

In order to increase the confidence of performance differences between user- and item-based recommender systems, further research should be done on how to deal with skewed item popularity. In addition, the performance differences between user- and item-based recommender systems should ideally also be investigated in OCCF settings that are less influenced by item popularity.

References

Agrawal, R., Srikant, R., et al. (1994). Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, volume 1215, pages 487–499.

Amatriain, X., Jaimes, A., Oliver, N., and Pujol, J. M. (2011). Data mining methods for recommender systems. In *Recommender Systems Handbook*, pages 39–71. Springer.

Bobadilla, J., Ortega, F., Hernando, A., and Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems*, 46:109 – 132.

Breese, J. S., Heckerman, D., and Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*.

Cho, Y. H., Kim, J. K., and Kim, S. H. (2002). A personalized recommender system based on web usage mining and decision tree induction. *Expert Systems with Applications*, 23(3):329–342.

Deshpande, M. and Karypis, G. (2004). Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1):143–177.

Goldberg, K., Roeder, T., Gupta, D., and Perkins, C. (2001). Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151.

Hassanat, A. B., Abbadi, M. A., Altarawneh, G. A., and Alhasanat, A. A. (2014). Solving the problem of the k parameter in the knn classifier using an ensemble learning approach. *International Journal of Computer Science and Information Security*, 12(8).

Hu, Y., Koren, Y., and Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272.

Instacart (2017). The instacart online grocery shopping dataset 2017. `https://www.instacart.com/datasets/grocery-shopping-2017`. Accessed: 2020-11-10.

Jalili, M., Ahmadian, S., Izadi, M., Moradi, P., and Salehi, M. (2018). Evaluating collaborative filtering recommender algorithms: A survey. *IEEE Access*, 6:74003–74024.

Jannach, D., Zanker, M., Felfernig, A., and Friedrich, G. (2010). *Recommender systems: An introduction*. Cambridge University Press.

Kamakura, W. A. (2008). Cross-selling: Offering the right product to the right customer at the right time. *Journal of Relationship Marketing*, 6(3-4):41–58.

Kamakura, W. A., Ramaswami, S. N., and Srivastava, R. K. (1991). Applying latent trait analysis in the evaluation of prospects for cross-selling of financial services. *International Journal of Research in Marketing*, 8(4):329–349.

Kohavi, R. et al. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada.

Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., and Riedl, J. (1997). Grouplens: Applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87.

Lin, W., Alvarez, S. A., and Ruiz, C. (2002). Efficient adaptive-support association rule mining for recommender systems. *Data Mining and Knowledge Discovery*, 6(1):83–105.

Miyahara, K. and Pazzani, M. J. (2000). Collaborative filtering with the simple bayesian classifier. In *Pacific Rim International Conference on Artificial Intelligence*, pages 679–689. Springer.

Mobasher, B., Dai, H., Luo, T., and Nakagawa, M. (2001). Effective personalization based on association rule discovery from web usage data. In *Proceedings of the 3rd International Workshop on Web Information and Data Management*, pages 9–15.

Ning, X., Desrosiers, C., and Karypis, G. (2015). A comprehensive survey of neighborhood-based recommendation methods. *Recommender Systems Handbook*, pages 37–76.

Pan, R., Zhou, Y., Cao, B., Liu, N. N., Lukose, R., Scholz, M., and Yang, Q. (2008). One-class collaborative filtering. In *2008 Eighth IEEE International Conference on Data Mining*, pages 502–511. IEEE.

Park, D. H., Kim, H. K., Choi, I. Y., and Kim, J. K. (2012). A literature review and classification of recommender systems research. *Expert Systems with Applications*, 39(11):10059 – 10072.

Paterek, A. (2007). Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD Cup and Workshop*, volume 2007, pages 5–8.

Ricci, F., Rokach, L., and Shapira, B. (2011). Introduction to recommender systems handbook. In *Recommender Systems Handbook*, pages 1–35. Springer.

Salakhutdinov, R., Mnih, A., and Hinton, G. (2007). Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th International Conference on Machine Learning*, pages 791–798.

Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, page 285–295, New York, NY, USA. Association for Computing Machinery.

Schafer, J. B., Frankowski, D., Herlocker, J., and Sen, S. (2007). *Collaborative Filtering Recommender Systems*, pages 291–324. Springer Berlin Heidelberg, Berlin, Heidelberg.

Zhou, L., Dai, L., and Zhang, D. (2007). Online shopping acceptance model-a critical survey of consumer factors in online shopping. *Journal of Electronic Commerce Research*, 8(1).