Recommendations on grocery shopping: customer or product similarities?

K.Y.M. de Kruiff
444677
Supervisor: F.J.L. van Maasakkers
Second assessor: S.I. Birbil

Bachelor Thesis: Econometrics and Operational Research
Erasmus School of Economics, ERASMUS UNIVERSITY ROTTERDAM

5th July 2019

Abstract

Online grocery shopping becomes more popular every day and benefits both customer and grocery store. To increase revenue, grocery stores recommend products to customers to add to their online baskets. Such recommendations can be either item- or consumer-based. In this research I investigate recommender systems based on the Collaborative Filtering algorithm. First, I replicate part of the paper of Li, Dias, Jarman, El-Deredy and Lisboa (2009), which investigates different standard item-based Collaborative Filtering models and introduces a new personalised recommender system, which outperforms the standard methods. Second, I investigate a user-based Collaborative Filtering model and compare these results to results of the item-based models. The results of the replication do not entirely agree with the results found by Li et al. (2009), which is mainly due to the higher number of items used in the replication. Despite the challenges of the user-based Collaborative Filtering model, the performance of the user-based model is similar to the performance of some of the item-based models, but does not outperform the item-based models. To generate all the results I have used a real-world dataset from the grocery store Ta-Feng.

The views stated in this thesis are those of the author and not necessarily those of Erasmus School of Economics or Erasmus University Rotterdam.

Contents

1 Introduction

Grocery shopping becomes easier every day. Nowadays a lot of brick-and-mortar grocery stores, i.e. Albert Heijn, Jumbo and Coop, have an online shopping website where you can order your groceries and have them delivered on your doorstep. While ordering groceries online, a grocery shop recommends products which are not yet in your shopping basket, by highlighting other products: 'You might also like...' or 'Did you forget...'. For example, if you have cake mix in your shopping basket but no butter or eggs, you probably get a recommendation for butter and eggs as these products are often sold with the cake mix. These recommendations are generated by recommender systems.

A good recommender system is beneficial for both customer and grocery store. A customer benefits from accurate recommendations, thereby diminishing the chance that crucial groceries do not get ordered. At the same time, accurate recommendations lead to more revenue for the grocery store as they will sell more products[1]. Therefore, a grocery store wants to optimise the recommender system used by its online webshop.

The main part of a recommender system is the personalisation algorithm that models consumer[2] shopping behaviour. This algorithm is used to search for items that are likely of interest to the individual customer. In general, the algorithm generates a rank-ordered list of items, which are not already present in the individual's basket. It is expected that a customer is more likely to accept recommendations on top of the list than items on the bottom of that list.

Several frameworks are used for item recommendations. For example, the Collaborative Filtering (CF) framework and the Random Walk model. These algorithms are used in a many papers (e.g. Breese, Heckerman & Kadie, 1998; Brin & Page, 1998; Sarwar, Karypis, Konstan, Riedl et al., 2001) that focus on recommendations for leisure products (e.g. movies, books and music). In contrast to products in the grocery shopping domain, leisure products are mostly purchased just once. In the domain of grocery shopping, customers buy products repeatedly. In addition, movies, books and music are often rated by their users, which is again not applicable to grocery shopping products. Many existing recommendation techniques are based on these user ratings. Hence, these techniques cannot simply be applied to grocery shopping data. Therefore, this research focuses on

[1] In this research, the terms products and items are used interchangeably when talking about recommender systems.
[2] In this research, the terms consumer and customer are used interchangeably.

the application of recommender systems for online grocery shopping. I investigate an algorithm introduced by Li et al. (2009) that appears to suit the characteristics of grocery shopping data better than the currently applied algorithms developed for leisure products. My research question is: How does the item-based algorithm introduced by Li et al. (2009) perform on grocery shopping data compared to a user-based CF algorithm? To answer my research question I first replicate part of the research of Li et al. (2009). Second, I investigate the user-based algorithm as described in Breese, Heckerman and Kadie (1998) for the same dataset as used in the replication and compare these results to the results of the replication.

The remainder of this research paper is organised as follows. Section 2 gives an overview of the empirical literature. A description of the retrospective data is given in Section 3. Section 4 describes the recommender systems used for the replication and the extension and the performance measures used to evaluate the recommender systems. The results of the replication and the extension are presented in Section 5. Finally, Section 6 provides the overall conclusion based on the results and Section 7 lists suggestions for improvement to this research and possibilities for future research.

2 Literature review

Previous research has demonstrated that CF is an effective framework to generate item recommendations (Breese et al., 1998; Konstan et al., 1997; Liu & Yang, 2008; Sarwar, Karypis, Konstan, Riedl et al., 2001; Yildirim & Krishnamoorthy, 2008). Researchers invented a number of CF algorithms that can be split into two categories: user-based and item-based algorithms (Breese et al., 1998).

User-based algorithms use the entire user-item database to generate a recommendation. These algorithms find a set of users, also called neighbours, that have the same preference (i.e. they either tend to buy a similar set of items or rate different products the same way) as the target user. The preferences of the neighbours are then used to generate one or more recommendations for the target user. This technique, known as nearest-neighbour or user-based CF, is very popular and widely used in practice. However, sparsity of the data and scalability of the process are challenges in user-based CF algorithms (Sarwar et al., 2001).

Item-based CF is a very simple method and has a relatively good performance. Therefore, it is one of the most popular recommendation algorithms. The item-based CF is a similarity-based algorithm, which assumes that consumers are likely to accept an item recommendation that is similar to an item bought previously. In this research, similarity is defined as how well a product supplements the products already present in the basket. Hence, the success of the item-based CF depends on the quality of the similarity measure. There are two types of similarity measures that are widely used in empirical literature (Li et al., 2009). First, the cosine-based measure, which is symmetric and determines the similarity between two products as the normalised inner product of two feature vectors of a user-item matrix. The feature vectors contain the number of times a user bought a certain product. To reduce the problem of overestimating the relevance of items that have an extremely high purchase frequency, the raw purchase frequency is converted to a log scale with $log(frequency + 1)$. Afterwards the $n \times m$ user-item matrix is normalised, where n is the number of users and m is the number of items. The cosine similarity is computed as:

$$sim(i,j) = \frac{M_{*,i} \cdot M_{*,j}}{|M_{*,i}||M_{*,j}|}, \tag{1}$$

where M reflects the $n \times m$ normalised user-item matrix and $M_{*,i}$ is the ith column of M.

Another similarity measure is the conditional probability-based similarity measure, which reflects the ratio between the number of users who have bought both items and the number of users who bought just one of the two items. This measure is asymmetric. A challenge of an asymmetric measure is that items often have high conditional probabilities with respect to very popular products caused by customers who purchase many items (Deshpande & Karypis, 2004). Therefore, Deshpande and Karypis (2004) propose a similarity measure that puts more emphasis on customers who have bought fewer items as these customers are more reliable indicators when determining the similarity. The measure is defined as:

$$sim(i,j) = \frac{\sum_{\forall q: M_{q,i} > 0} M_{q,j}}{Freq(i) \times (Freq(j))^{\alpha}}, \tag{2}$$

where $\alpha \in [0,1]$ and is used to penalise very popular products, $Freq(i)$ the number of users that have purchased item i, and M the same normalised $n \times m$ user-item matrix as used in (1). As (2) uses the non-zero entries of the normalised user-item matrix M, customers contribute less to the total similarity if they purchased more items. Hence, more emphasis is put on the purchasing decisions of the customers who bought fewer items (Deshpande & Karypis, 2004).

After determining the similarity matrices, the recommendations are determined based on the products already in the customer's basket. The rows of the similarity matrix belonging to the products already in the customer's basket are added together, which results in a single row matrix containing the total similarity of each possible item. Then, the items with the highest values in the resulting row matrix form a list of recommendations for the user.

Previous research has shown that item-based CF is able to achieve recommendation accuracies similar to, or better than, the user-based approach (Sarwar et al., 2001). However, a challenge of the item-based CF is that it is not able to recommend an item that has never been co-purchased with the other items before. A solution to this deficiency is to use a Random Walk based recommendation algorithm, which randomly jumps from product to product.

Besides the different CF algorithms, recommender systems can be based on a Random Walk which will be described in Section 4. Further, there are several machine learning techniques that can also be applied to generate recommendations for customers, i.e. the nearest-neighbour technique and the Random-Forest framework. The machine learning techniques are beyond the scope of this research.

3 Data

In this research I use a basket dataset from the grocery store Ta-Feng.[3] Ta-Feng is a Chinese membership retailer warehouse that sells a wide variety of goods (Hsu, Chung & Huang, 2004). The dataset consists of shopping records of different customers from November 2000 until February 2001 and contains, among other data, the shopping date, customer ID, type of product and the amount of the product bought. If several shopping records have the same shopping date and the same customer ID, they are seen as one transaction. To determine the number of transactions, I order the data on shopping date and afterwards on customer ID. For evaluation purposes I exclude the records from customers that only ordered products from Ta-Feng once in the four month period. The resulting dataset consists of 20,388 customers, 107,700 transactions and 2,000 product sub-classes. The sparsity, which represents the ratio of empty entries in the co-occurrence matrix, is 0.625. The other descriptive statistics of the resulting dataset are given in Table 1.

After deleting all customers with only one transaction during the four month period, I split the data into two parts: a training and a test set. The test set consists of the last transaction of each customer and the training set consists of all other transactions aggregated per customer into one basket to maximise the information content. Some of the product sub-classes are never purchased in the training set and are therefore excluded, which results in 1,973 remaining sub-classes. From the test set, I delete the transactions containing less than four items for evaluation purposes. This

[3] https://www.kaggle.com/chiranjivdas09/ta-feng-grocery-dataset

results in a training set of 87,312 transactions and a test set of 11,813 transactions.

Table 1: Descriptive statistics adjusted Ta-Feng dataset

	Min.	Median	Mean	Max.
# products per basket	1	6	9.24	1,200
# products per sublcass	1	57	497.81	21,744
# baskets per customer	2	4	5.28	86

4 Methodology

In this section, I describe the methods that I apply to answer my research question stated in the introduction. Section 4.1 describes the basic Random Walk model and Section 4.2 describes the personalised Basket-Sensitive Random Walk introduced by Li et al. (2009). The performance measures used for evaluation of the recommender systems are explained in Section 4.3 and Section 4.4 describes the user-based CF model.

4.1 Random Walk model

Many researchers have used a Random Walk model to generate recommendations for movies (Liu & Yang, 2008; Yildirim & Krishnamoorthy, 2008; Wijaya & Bressan, 2008; Huang, Zeng & Chen, 2004). The Random Walk model is very similar to Google's PageRank algorithm, which is a link analysis algorithm that assigns an importance rank to each page. The importance rank of a page reflects the probability of visiting that page in a Random Walk (Brin & Page, 1998). Yildirim and Krishnamoorthy (2008) use this idea to develop an item graph where the nodes are items and the edges reflect the similarities between the items. The algorithm introduced by Yildirim and Krishnamoorthy (2008) generates a user-item matrix, R_{basket}, which contains the ratings of the items for each user and is computed as follows:

$$R_{basket} = MdP(I - dP)^{-1}, \tag{3}$$

where M is the same user-item matrix as in (1) and (2), P is the transition probability matrix and $d \in (0, 1)$ is the damping factor used to define the stopping probability of the Random Walk. There are two possibilities for the transition matrix P, namely the normalised cosine and the normalised conditional probability-based similarity matrix. The normalisation of the matrices is needed to derive a transition probability matrix from the similarity matrices. The products with the highest rating in R_{basket} that are not yet contained in a customer's basket, are recommended to the customer. Thus, the Random Walk model is personalised but does not compute the product ratings based on the products already in the basket.

4.2 Basket-Sensitive Random Walk on bipartite network

As described in Section 4.1, the Random Walk model does not take the current basket into account when recommending new products. Therefore, Li et al. (2009) introduce the Basket-Sensitive Random Walk algorithm. This approach computes the transaction probabilities between products (not users) from a bipartite network as described in Section 4.2.1. Further, the model computes the ratings of the products while taking into account the products already in the basket. In this way, relevant products are given a higher rating on the list of recommendations, with less bias to the most popular products.

4.2.1 First-Order Transition Probability Matrix

The shopping basket data can be seen as a bipartite network with two types of nodes: consumers and products. Each edge, being between a consumer and a product, reflects the consumer's purchase frequency of a product as shown in Figure 1.

The set of consumers is denoted by $C = c_1, c_2, ..., c_{|C|}$, the set of products by $P = p_1, p_2, ..., p_{|P|}$ and the set of purchase frequencies by $F = f(1,1), f(1,2), ..., f(|C|, |P|)$, which results in the bipartite network $BN = \{C, P, F\}$. An example of a bipartite network is given in Figure 1.

In a Random Walk, the first-order transition probability $P(p_j|p_i)$ is the probability that a random surfer jumps from the item node i to j via all connected consumer nodes c_k. As the Random Walk only jumps via consumers who bought *both* products, the transition matrix consists of merely *first-order* transition probabilities which can be written as:

$$P(p_j|p_i) = \sum_{k=1}^{|C|} P(p_j|c_k)P(c_k|p_i), \tag{4}$$

where $P(p_j|c_k)$ is the probability that a random surfer jumps from consumer node c_k to product node p_j, $P(c_k|p_i)$ the probability that the random surfer jumps to consumer node c_k from the product node p_i and $P(p_j|p_i)$ denotes the marginal probability distribution over all consumers which reflects the similarity between products i and j. The intuition behind probability $P(p_j|p_i)$ is that it can be seen as the preference for product p_j from all customers C who have already bought product p_i. Every preference $P(p_j|c_k)$ from the kth customer is weighed proportionally to the number of times the customer bought product p_i, namely $P(c_k|p_i)$. The conditional probabilities used in (4) are calculated as:

$$P(p_j|c_k) = \frac{f(c_k, p_j)}{(\sum f(c_k, \cdot))^{\alpha_1}}, \quad (5) \qquad P(c_k|p_i) = \frac{f(c_k, p_i)}{(\sum f(\cdot, p_i))^{\alpha_2}}. \quad (6)$$

where $\alpha_1, \alpha_2 \in [0, 1]$ correct for products that have high purchases frequencies. For convenience reasons this research applies $\alpha_1 = \alpha_2$. As in (2), the parameter α corrects for the fact that people who buy a lot of different products are less informative than people who bought just a few products. Also, more popular products are less informative for the personal preferences of the shoppers than unpopular products. However, the first-order similarity measure does not capture the possible similarity between products which are never bought together, like newly launched products (Li et al., 2009). Hence, when data is sparse, the first-order similarity measure suffers and results in less effective recommendation.

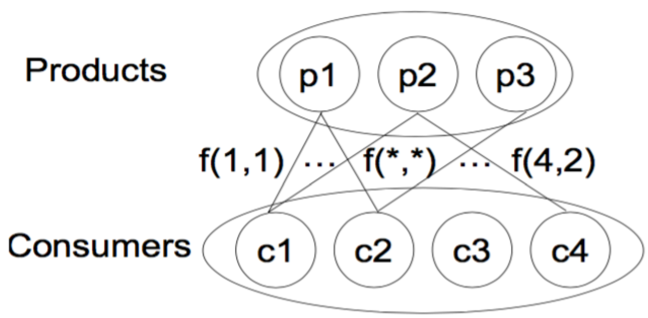


Figure 1: Bipartite network for shopping basket data where $f(i, j)$ is the amount of product j bought by customer i (Li et al., 2009)

4.2.2 Basket-Sensitive Random Walk

As described in Section 4.2.1, the transition probability matrix has a disadvantage when data is sparse. This limitation can be solved by combining the similarities of all orders as:

$$P^* = \sum_{t=1}^{\infty} \frac{(dP)^t}{|(dP)^t|}, \quad (7)$$

where P is either the cosine or conditional probability matrix described in **??** or the first-order transition probability matrix explained in 4.2.1. Again, the parameter $d \in (0, 1)$ is the damping factor which determines the length of the Random Walk. When t, the number of purchases of a product, goes to infinity, P^* converges to $dP(I - dP)^{-1}$ (Yildirim & Krishnamoorthy, 2008).

As the Random Walk P^* is not based on the current shopping behaviour, the model has to be biased towards the items already in the shopping basket in order to get more accurate results. To do so, an additional rating can be assigned to items currently in the shopper's basket during each iteration of the Random Walk computation. Therefore, Li et al. (2009) introduce a non-uniform personalisation vector U_{basket} to compute the importance score during each iteration as follows:

$$R_{basket} = d \cdot P \cdot R_{basket} + (1 - d) \cdot U_{basket}, \tag{8}$$

where R_{basket} contains the ratings for all products derived from the basket-based importance scores and $U^i_{basket} = \frac{1}{m}$ if the i^{th} product is in the basket and zero otherwise, where m is the number of products in the current basket. The matrix P can be the normalised cosine or conditional probability-based similarity matrix or the first-order transition probability matrix as described in Section 4.2.1. The first term of (8), $d \cdot P \cdot R_{basket}$, can be interpreted as the probability that the Random Walk stops and that the customer is done shopping. The second term of the equation, $(1 - d) \cdot U_{basket}$, can be seen as the probability that the Random Walk proceeds, meaning that the customer is still adding items to the basket. However, applying (8) in practice would be too time consuming due to the re-computations of the ratings in each iteration of the Random Walk. Therefore, Li et al. (2009) also proposed an approximation, $\hat{R}_{basket}$ defined as:

$$\hat{R}_{basket} = \sum_{p_i \in basket} R_{item_i}, \tag{9}$$

$$R_{item_i} = d \cdot P \cdot R_{item_i} + (1 - d) \cdot U_{item_i} \tag{10}$$

where R_{item} is the item-based importance score and U_{item_i} is the personalisation vector with the i^{th} entry set to one, and the rest set to zero. The computation of R_{item_i} can be simplified to

$$R_{item_i} = (1 - d)(I - dP)^{-1} U_{item_i}, \tag{11}$$

which also simplifies the computation of (9). The products with the highest rating in $\hat{R}_{basket}$, which are not already present in the basket, are recommended to the consumer.

4.3 Performance metrics and evaluation protocols

There are several ways to evaluate the accuracy of a recommender system. It is important that the performance evaluation results are representative of live, interactive behaviour. Sordo-Garcia, Dias, Li, El-Deredy and Lisboa (2007) looked into three evaluation strategies which differed in how to split the retrospective basket into evidence and target components, where the evidence products are used to predict the target products. The three approaches were to split the data randomly, based on popularity and via the leave-n-out approach (Breese et al., 1998). According to the results the popularity-based approach was the only one that ranked the recommender systems consistently with their live performance (Sordo-Garcia et al., 2007).

To evaluate the accuracy of the recommender systems, this research uses the popularity-based binary hit rate, `bHR(pop)`, and the weighed hit rate, `wHR(loo)`, introduced by Li et al. (2009). The popularity based binary hit rate is based on a leave-three-out principle; the least three popular products of the test basket, based on the training set, are the targets, the rest of the test basket is used as evidence. When testing the recommender systems, the evidence items are used to predict the target items. The binary hit rate is computed as the proportion of test baskets having at least one out of three correctly predicted target items (Li et al., 2009). Besides leaving out the three least popular items, I also evaluate the performance of the recommender system when using a random leave-three-out approach (`bHR(rnd)`), where three target items are randomly selected from the test basket. However, `bHR(rnd)` favors items which occur with a high frequency and therefore tends to overestimate the model performance. Moreover, previous research has shown that when using a leave-one-out cross validation the hit rates also over-emphasize the performance of popular products (Sordo-Garcia et al., 2007; Li, Dias, El-Deredy & Lisboa, 2007).

To solve the issue of over-estimation for a leave-one-out cross validation, Li et al. (2009) propose a new performance measure, which weighs the hit of items inversely to their popularity. The weighed hit rate based on a leave-one-out approach, `wHR(loo)`, is computed as:

$$\texttt{wHR(loo)} = \frac{\sum_i (1-p(x_i))\dot{H}IT(x_i)}{\sum_i (1-p(x_i))} \quad \text{with} \quad HIT(x_i) = \begin{cases} 1, & \text{if } x_i \text{ is predicted correctly} \\ 0, & \text{otherwise} \end{cases}, \tag{12}$$

where x_i is the target item and $p(x_i)$ is its prior probability based on the popularity of the product. The popularity and hence the prior probability $p(x_i)$ is based on the converted user-item matrix and is computed as the ratio between the number of times an item was bought and the total number of items bought by all customers. When all items are predicted correctly, the basket will achieve a hit rate of one. The final hit rate is computed by averaging over all test baskets. Another measure based on a leave-one-out cross validation that biases the results towards the performance of small classes, is the macro-average hit rate, `macroHR(loo)`. The `macroHR(loo)` measure computes the hit rate for each product in a basket, which results in a vector of zeros, reflecting no hit, and ones, reflecting a hit. The final hit rate is averaged over all products, which is the length of the vector, instead of all baskets as for the `wHR(loo)`.

In total I evaluate three types of standard item-based CF models, the cosine (1), the conditional probability (2) and the bipartite network (4) based similarities, referred to as *cf(cos)*, *cf(cp)* and *cf(bn)* respectively. I also evaluate the standard Random Walk item-based CF model (*rw*) described in Section 4.1 and the Basket-Sensitive Random Walk model (*bsrw*) described in Section 4.2. All these methods are item-based models and are compared to a baseline method, *pop*, which merely recommends the most popular items not contained in the basket.

4.4 User-based Collaborative Algorithm

Besides the item-based CF algorithm, I investigate the user-based CF algorithm for the same Ta-Feng dataset, which is not yet done by Li et al. (2009). The idea of the user-based algorithm is to predict the ratings of the products for a particular user, also referred to as active user, based on the user-ratings of similar users from the training set, where ratings are based on the number of purchases of a product. The user database for the Ta-Feng dataset is not too big, namely 20,388 customers. Hence, the scalability problem, which was mentioned in Section 2, of the user-based CF algorithm might be avoided. Therefore, it is interesting to investigate whether for the size of this user database a user-based CF approach could also work, despite its 'cold start' problem and scalability issues (Konstan et al., 1997; Shardanand & Maes, 1995).

The user-based CF algorithm makes use of the cosine similarity matrix containing the similarities between users instead of items. Again, the normalised $n \times m$ user-item matrix M is constructed as described in Section 2 and can be used to compute the cosine similarity matrix for the users as:

$$sim(i,j) = \frac{M_{i,*} \cdot M_{j,*}}{|M_{i,*}||M_{j,*}|}, \tag{13}$$

where $M_{i,*}$ is the ith row of the $n \times m$ user-item matrix M representing the quantity of the items bought by customer i (Deshpande & Karypis, 2004).

Before predicting ratings for the active user for the test set, the k most similar users are determined from the cosine similarity matrix. Then, the predicted ratings of the active user for item j,

$r_{a,j}$, are computed as a weighed sum of the ratings of the k most similar users as:

$$r_{a,j} = \bar{v}_a + \sum_{i=1}^{k} w(a,i)(v_{i,j} - \bar{v}_i), \tag{14}$$

where $\bar{v}_i$ is the mean of all the ratings of the products in the basket of customer i for the training set, $v_{i,j}$ is the rating from user i for item j and $w(a,i)$ is the normalised cosine similarity between user a and i (Breese et al., 1998). The mean of all ratings is computed as:

$$\bar{v}_i = \frac{1}{|I_i|} \sum_{j \in I_i} v_{i,j}, \tag{15}$$

where I_i is the set of items of user i for which there exists a rating in the training set. As for the item-based algorithm, the user-based algorithm is evaluated with the test set which is split into targets and evidence. With the targets and evidence products, the performance metrics `bHR(pop)`, `bHR(rnd)` and `wHR(loo)` can be computed based on the exact same training and test set as used to generate the replication results. In this way an accurate comparison can be made to the results of the item-based CF models. However, this also makes it hard to compare the results of the user-based CF model to the results of Li et al. (2009) as this research makes use of a different composition of the Ta-Feng dataset.

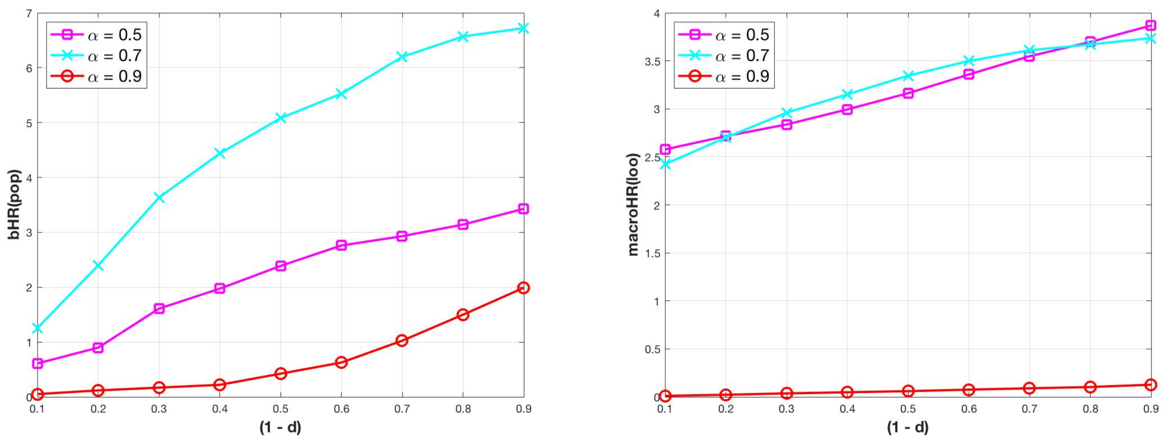
Besides evaluating the performance of the user-based CF algorithm compared to the item-based CF algorithm and the baseline method *pop*, I intend to find an optimal level for k. To determine the optimal level for k I investigate a trade-off between the values of the performance metrics and the computation time of the user-based CF algorithm. For the computation time I focus on the time needed to compute (14) as this will differ most between the different values for k. Suggestions for an optimal value for k are widely varying in empirical literature (Mobasher, Dai, Luo & Nakagawa, 2001; Sarwar et al., 2001; Al Mamunur Rashid, Karypis & Riedl, 2006). Therefore, I consider values going from three to 1000.

5 Results

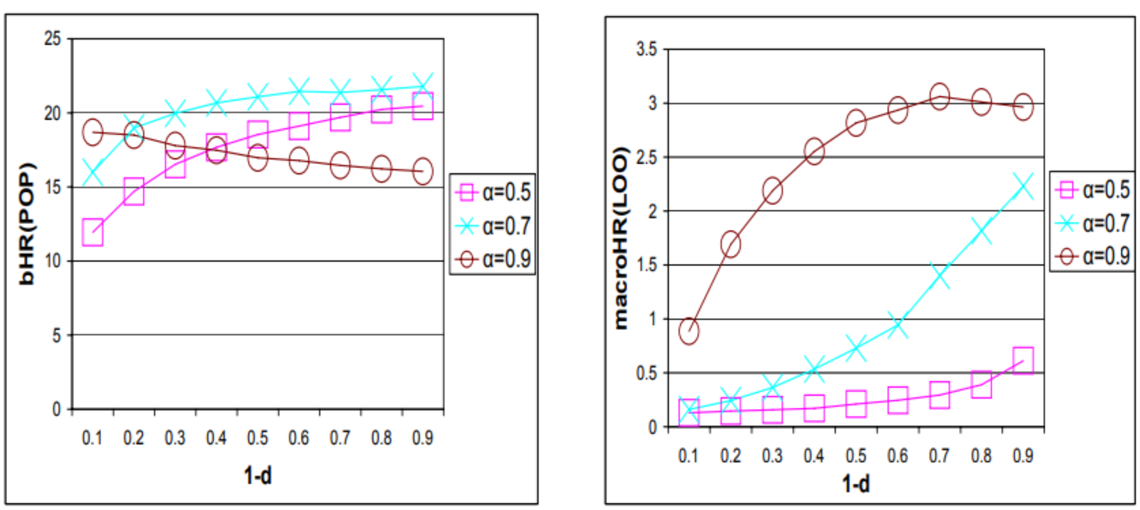
This section gives an overview of the results of the replication of Li et al. (2009) in paragraph 5.1 and paragraph 5.2 gives the results of the user-based CF algorithm and a comparison to the results of the replication.

5.1 Results replication

The results of the replication in this paper are similar to the results obtained by Li et al. (2009), but not exactly the same. Figure 2 shows the performance of the metrics `bHR(pop)` and `macroHR(loo)` with respect to the parameters α and d for the results of the replication (Table 2a) and for the results of Li et al. (2009) (Table 2). Table 2 reports the best performance of the models with respect to the penalty factor α, and the damping factor d for the measures `bHR(pop)`, `bHR(rnd)` and `wHR(loo)` described in Section 4.3.



(a) Replication results for *cf(bn)+bsrw*



(b) Plot result Li, Dias, Jarman, El-Deredy and Lisboa (2009) for *cf(bn)+bsrw*

Figure 2: Predictive performance for different settings of α and d for the Basket-Sensitive Random Walk model

It is expected that the performance measures increase when the parameter values increase. Namely, if α increases, the similarity between popular items decreases and if d decreases, so $(1-d)$ increases, then the ratings increase for those similar but less frequent items (Li et al., 2009). However, this does not hold for the results of the replication in this paper, which can be seen from Figure 2a. Moreover, this expectation does not hold for the results of Li et al. (2009) either. Figure 2a shows that the performance measures do increase when $(1-d)$ increases, but they decrease when α increases. The difference with the result of Li et al. (2009) can be explained by the fact that when α and $(1-d)$ increase, less frequent items appear more often on top of the list of recommendations. However, Li et al. (2009) use 1,093 items and the replication 1,973 items. Hence, when the parameters increase, the probability of correctly predicting the least frequent items is smaller for the replication than for Li et al. (2009), which causes the inconsistencies seen in Figure 2a. The inconsistencies of the results of Li et al. (2009) can be explained by the fact that the macro-averaged measure (`macroHR(loo)`) is biased to performance in small classes compared to the basket-averaged measure (`bHR(pop)`). Hence, `macroHR(loo)` increases when more infrequent items are predicted correctly and `bHR(pop)` decreases as popular items are probably missed (Li et al., 2009).

Based on Table 2, some remarks can be made on the results of the replication with respect to the results generated by Li et al. (2009). First of all, Table 2a shows that almost all methods outperform the baseline method, *pop*, which is in line with the results of Li et al. (2009) presented in Table 2b. Second, it is remarkable to see that for almost every method the values of the binary hit rate based on popularity of the replication are significantly smaller than the results generated by Li et al. (2009) except for the Random Walk method. This difference can again be explained by the fact that this research considers 1,973 items for the customers to choose from instead of 1,093 as in the study of Li et al. (2009). With more items, the probability of predicting the least popular items correctly becomes smaller and therefore the binary hit rate based on popularity decreases.

Third, from Li et al. (2009) it was not entirely clear whether they performed a personalised or a non-personalised Random Walk algorithm. The Random Walk model applied in this research is personalised as it includes a multiplication with the user-item matrix as described in Section 4.1. The personalisation of the Random Walk model could explain the relatively high results for the performance metrics for the Random Walk model. Hence, Li et al. (2009) probably applied a non-personalised Random Walk model which explains the difference in results. However, it should be taken into consideration that the number of items used differs and could also be of influence.

Further, as mentioned in Section 4.3, `bHR(rnd)` favors very popular items and therefore tends to overestimate the model performance. This is in line with the results of the replication and the results of Li et al. (2009) as the values for the `bHR(rnd)` metric are higher than for the `bHR(pop)` metric.

Finally, the results for the Basket-Sensitive Random Walk of the replication are lower than the results generated by Li et al. (2009) for the `bHR(pop)` measure, which is already explained above. The rest of the evaluation measures generate more or less the same results as Li et al. (2009) do. However, the results for the Basket-Sensitive Random Walk model based on the bipartite network transition probability matrix for the replication are not as outstanding as the results of Li et al. (2009). This could again be due to the number of items used in the replication or a misinterpretation of the method as described in Li et al. (2009). When only comparing the results of the Basket-Sensitive Random Walk model to the results of the standard CF model, the *cf(bn) + bsrw* method is always among the top two performers. Also, the bipartite network-based Basket-Sensitive Random Walk model has the highest hit rates compared with the cosine and conditional probability-based Basket-Sensitive Random Walk model, which is in line with the results of Li et al. (2009). This can be explained by the fact that the transition probability of the bipartite network is directly structured from the graph and hence is more effective than the transition probability indirectly derived by normalising the similarity matrices (Li et al., 2009).

Table 2: Predictive performance comparison of different models

(a) Results replication (%)

	L-3-O		**L-1-O CV**
Method	`bHR(pop)`	`bHR(rnd)`	`wHR(loo)`
cf(cos)	6.2728	20.7483	**4.3438**
cf(cp)	4.7067	21.0023	4.1406
cf(bn)	5.4770	**24.1260**	2.4651
cf(cos) + bsrw	6.7891	18.6405	3.2600
cf(cp) + bsrw	4.5374	20.7822	4.1205
cf(bn) + bsrw	6.7214	24.0159	4.3078
cf(cos) + rw	11.1149	23.8551	3.9529
cf(cp) + rw	**12.6132**	19.9611	2.9326
pop	0.5418	18.3357	2.4944

(b) Results Li et al. (2009) (%)

	L-3-O		**L-1-O CV**
Method	`bHR(pop)`	`bHR(rnd)`	`wHR(loo)`
cf(cos)	18.1945	22.0730	3.9281
cf(cp)	14.6064	19.9129	3.3541
cf(bn)	21.7341	25.2556	4.6646
cf(cos) + bsrw	18.2429	22.6599	3.9652
cf(cp) + bsrw	16.7605	20.6571	3.4367
cf(bn) + bsrw	**21.7886**	**26.1814**	**4.8151**
cf(cos) + rw	7.7994	5.4880	0.7289
cf(cp) + rw	14.0679	19.1807	3.1493
pop	7.9900	16.5700	2.2800

Note: For the results of the replication the metric based on random selection of targets is evaluated just once as it did not differ much per run and spared computation time.

5.2 Results user-based CF algorithm

Table 3 shows the results for the user-based CF model for different values of k, which reflects the number of users used to determine the ratings of the active user. Further, Table 3 shows that the user-based CF outperforms the baseline method, *pop*, denoted in Table 2a for all values of k. The user-based CF also outperforms the standard item-based CF model based on the leave-three-out metrics for almost every k. However, the best methods for the `bHR(pop)` metric of the Random Walk model and Basket-Sensitive Random Walk model shown in Table 2a outperform the user-based CF model, but for the `bHR(rnd)` metric the user-based CF model is better for $k \geq 20$. The above results are based on the results for different values of k for the test set. However, for a more accurate comparison between the results of the item- and user-based CF model, a validation set should be used to determine the optimal level for k first and afterwards the results of the performance metrics should be determined for the test set for the optimal value of k. These latter results should then be used for comparison with the results of the item-based CF model.

Besides comparing the user-based CF model to the item-based CF models, it is interesting to see what the optimal value for k is. The evaluation of the performance metrics with respect to k is shown in Figure 3. The exact values of the performance metrics displayed in Figure 3 correspond to the results denoted in Table 3. Table 3 and Figure 3 show that the macro-averaged metric does not differ much amongst different values of k. This could be explained by the fact that `macroHR(loo)` is based on a leave-one-out cross validation and hence that the recommended item with the highest rating does not differ much when taking more customers into consideration. The basket-averaged binary hit rate based on popularity decreases once k increases. This can be due to the number of items taken into consideration determining the ratings. When k increases, more and more users are used who are not very similar to the active user. This leads to less accurate rating prediction as a lot of irrelevant information is used and more products are probably considered, which causes the `bHR(pop)` to decrease. Furthermore, the random basket-averaged binary hit rate and the elapsed time both increase when k increases. As seen before, the `bHR(rnd)` tends to overestimate the model performance as it favors very popular products. When using more information as k increases, the very popular products will probably appear more often, which causes overestimation of the metric.

Depending on what metric is important for a research, one could determine the optimal value for k. However, I believe that $k = 10$ can be seen as the most appropriate value for k for this dataset as

Table 3: Predictive performance user-based CF algorithm for different sizes of neighbourhood

	L-3-O (%)		**L-1-O CV (%)**	
Method	**bHR(pop)**	**bHR(rnd)**	**macroHR(loo)**	**Elapsed time (sec)**
k = 3	**10.5223**	22.6191	3.5400	**32.4475**
k = 5	10.1840	23.3472	3.6800	38.8772
k = 10	9.7858	23.9228	3.7500	47.8670
k = 20	9.5149	24.3799	3.8700	53.1561
k = 50	9.1425	25.1926	3.9800	87.7133
k = 100	8.6346	24.5916	**4.0200**	146.1390
k = 500	7.4410	25.6328	3.9700	728.6756
k = 1000	6.8315	**26.4708**	3.9300	1083.4911

Note: The metric based on random selection of targets is evaluated just once as it did not differ much per run and spared computation time.

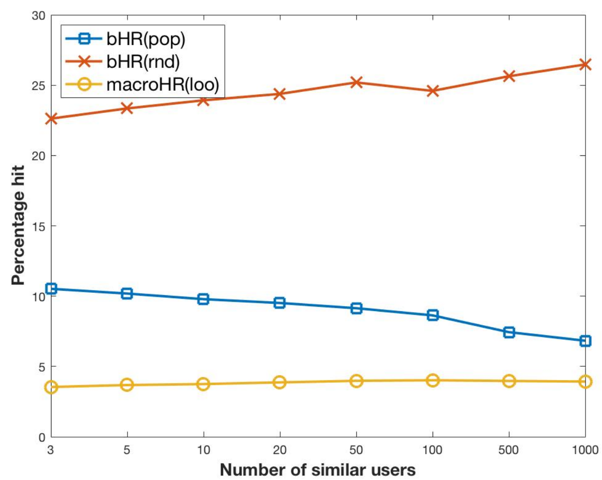


Figure 3: Evaluation of different metrics for different sizes of neighbourhood

the computation time for the ratings is less than one minute, the metric `bHR(pop)` is still relatively high and the measure `macroHR(loo)` does not increase much when increasing the value of k. As the `bHR(rnd)` is probably overestimated even more for a higher value of k, a lower value of k is also preferable for `bHR(rnd)`.

6 Conclusion

Grocery stores implement a variety of recommender systems to recommend products that are not already present in a customer's basket. As it benefits both customer and grocery store to have the best possible working recommender system, this research investigates a basket-sensitive recommender system introduced by Li et al. (2009). I replicate the paper from Li et al. (2009) for the Ta-Feng dataset and apply a user-based CF model to the same dataset. The results of the user-based CF model are compared to the results of the item-based models. Furthermore, I assess into the optimal number of users used to determine the ratings for the active user.

The results of the replication of Li et al. (2009) in this paper approach the results of Li et al. (2009) to a large extent. However, with a different set of items used in this paper for the replication, the replication does not yield the exact same conclusion as Li et al. (2009). Li et al. (2009) conclude that the Basket-Sensitive Random Walk model performs better than all other models they investigated. However, this paper finds that the Basket-Sensitive Random Walk model can be outperformed by some of the other models. The results of the replication show that every performance metric has a different optimal method, but that overall *cf(bn) + bsrw* performs well on all measures.

The user-based CF model outperforms the standard item-based CF model, but in turn outperformed by some of the Random Walk and Basket-Sensitive Random Walk models. This can be explained by the fact that the user-based CF model does not take into account the items already in the customer's basket, whereas the Random Walk and Basket-Sensitive Random Walk model do. Overall, the Basket-Sensitive Random Walk is the most appropriate recommender system in the domain of grocery shopping as the difference with the best performing models per metric is very small and the model is among the top two performers for every metric, which is not the case for the other models.

Finally, the optimal number of neighbours used to determine the ratings of the active user is determined by examining the three different hit rates, `bHR(pop)`, `bHR(rnd)` and `macroHR(loo)`, and by elapsed time for computing the ratings of the active user. For this research the optimal number of neighbours used is ten, based on a trade-off between the decrease of `bHR(pop)` and an increase in `bHR(rnd)` and the computation time.

7 Discussion

As some of the results of the replication do not agree with the results generated by Li et al. (2009) there is room for improvement to this research. One of the big differences between the replication and Li et al. (2009) is the number of items used as it was not clear from Li et al. (2009) how they determined the number of items used in their research. Therefore, it would be interesting to determine whether the results are more alike when taking the 1,093 most popular items and base the replication on only those 1,093 items. From Li et al. (2009) it is not clear if they took the 1,093 most popular items. However, with further investigation of the results, this might become clear. Further, the results for the popularity based binary hit rate based on 1,093 items will show whether the low hit rates for `bHR(pop)` are due to the number of items or whether there is a different underlying reason.

There are several other limitations to this research. First of all, in this research the items are all organised in sub-classes. This is probably done to overcome the sparsity and scalability problem, but it also causes a loss of information. However, a sub-class item recommendation is not as precise as a specific product recommendation and could therefore not be optimal. Second, people are price-sensitive and this research does not take the price of products into consideration while generating recommendations. Not everyone has the same income, and therefore the same budget, to do groceries. The price of a product is a really important factor and by including the price factor in the recommender system, recommendations can be more accurate. Furthermore, it is also of interest for the grocery stores to recommend products with a high profit margin. Hence, including this in the recommender system benefits the grocery store. Third, there are a lot of other factors that may also be of interest when generating recommendations for grocery shoppers, which are not yet included in this paper, i.e. age of the customer, location of product in the store, and whether it is a seasonal product yes or no. Hence, future research needs to point out how including more details in the recommender system benefits the performance of recommender systems.

Besides the ideas for future research proposed by Li et al. (2009), I think it is interesting to investigate the user-based approach in more detail. There are several other methods that can be applied to determine similarity between customers. For example, it can be interesting to cluster customers by age, average price of the basket and average size of the basket and provide recommendations based on the cluster to which a customer belongs. Furthermore, it is also interesting to examine how machine learning techniques like Random-Forest and nearest-neighbour perform on grocery recommendations. These machine learning techniques are, just like the CF algorithm, widely explored in the field of movie recommendation, but have not yet been applied on the domain of grocery shopping. Besides, it might even be possible to apply an item-based machine learning approach to the dataset. One of the main advantages of machine learning techniques is that they learn from their ‘mistakes’. By applying the machine learning technique on a greater amount of data, the machine learning technique is able to recognise patterns and provide even more accurate recommendations.

References

Al Mamunur Rashid, S. K. L., Karypis, G. & Riedl, J. (2006). Clustknn: A highly scalable hybrid model-& memory-based cf algorithm. *Proceeding of webKDD, 2006*.

Breese, J. S., Heckerman, D. & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the fourteenth conference on uncertainty in artificial intelligence* (pp. 43–52). Morgan Kaufmann Publishers Inc.

Brin, S. & Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, *30*(1-7), 107–117.

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

Hsu, C.-N., Chung, H.-H. & Huang, H.-S. (2004). Mining skewed and sparse transaction data for personalized shopping recommendation. *Machine Learning*, *57*(1-2), 35–59.

Huang, Z., Zeng, D. & Chen, H. (2004). A link analysis approach to recommendation under sparse data. *AMCIS 2004 Proceedings*, 239.

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

Li, M., Dias, B. M., Jarman, I., El-Deredy, W. & Lisboa, P. J. (2009). Grocery shopping recommendations based on basket-sensitive random walk. In *Proceedings of the 15th acm sigkdd international conference on knowledge discovery and data mining* (pp. 1215–1224). ACM.

Li, M., Dias, B., El-Deredy, W. & Lisboa, P. J. (2007). A probabilistic model for item-based recommender systems. In *Proceedings of the 2007 acm conference on recommender systems* (pp. 129–132). ACM.

Liu, N. N. & Yang, Q. (2008). Eigenrank: A ranking-oriented approach to collaborative filtering. In *Proceedings of the 31st annual international acm sigir conference on research and development in information retrieval* (pp. 83–90). ACM.

Mobasher, B., Dai, H., Luo, T. & 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* (pp. 9–15). ACM.

Sarwar, B. M., Karypis, G., Konstan, J. A., Riedl, J. et al. (2001). Item-based collaborative filtering recommendation algorithms. *Www*, *1*, 285–295.

Shardanand, U. & Maes, P. (1995). Social information filtering: Algorithms for automating

. In *Chi* (Vol. 95, pp. 210–217). Citeseer.

Sordo-Garcia, C. M., Dias, M. B., Li, M., El-Deredy, W. & Lisboa, P. J. (2007). Evaluating retail recommender systems via retrospective data: Lessons learnt from a live-intervention study. In *Dmin* (Vol. 95, pp. 197–206). Citeseer.

Wijaya, D. T. & Bressan, S. (2008). A random walk on the red carpet: Rating movies with user reviews and pagerank. In *Proceedings of the 17th acm conference on information and knowledge management* (Vol. 95, pp. 951–960). ACM.

Yildirim, H. & Krishnamoorthy, M. S. (2008). A random walk method for alleviating the sparsity problem in collaborative filtering. In *Proceedings of the 2008 acm conference on recommender systems* (Vol. 95, pp. 131–138). ACM.

A Matlab code

Listing 1: Code to generate the descriptive statistics for the Ta-Feng dataset

```
transactions2 = 1;
baskets2 = [];
transDate2 = [];
customerTransaction2 = [];
subclassPerBasket = zeros(107700,2000);
indexNewBasket = [];
%amount of baskets
for i = 1:720164
    if i == 1
        baskets2(transactions2) = newAmount(i);
        transDate2(transactions2) = newTransactionDate(i);
        customerTransaction2(transactions2) = newCustomerID(i);
        indexNewBasket = [indexNewBasket; i];
    else if newTransactionDate(i+1) == newTransactionDate(i) && newCustomerID(i+1)
       == newCustomerID(i)
            baskets2(transactions2) = baskets2(transactions2) + newAmount(i+1);
        else
            indexNewBasket = [indexNewBasket; i+1];
            transactions2 = transactions2 + 1;
            baskets2(transactions2) = newAmount(i+1);
            transDate2(transactions2) = newTransactionDate(i+1);
            customerTransaction2(transactions2) = newCustomerID(i+1);
        end
    end
end
meanBasket = mean(baskets2);
medianBasket = median(baskets2);
minBasket = min(baskets2);
maxBasket = max(baskets2);
%amount of customers
[sortedCustomers2, ia, indexSortedCustomers2] = unique(customerTransaction2);
%amount of subclasses
[sortedSubClasses2, is, indexSortedSubclasses2] = unique(newSubClasses);
%amount of products per subclass
amountPerSubclass = zeros(2000, 1);
for j=1:2000
    for i = 1:size(newAmount,2)
        if indexSortedSubclasses2(i) == j
            amountPerSubclass(j) = amountPerSubclass(j) +newAmount(i);
        end
    end
end
meanAmountSC = mean(amountPerSubclass);
medianAmountSC = median(amountPerSubclass);
maxAmountSC = max(amountPerSubclass);
minAmountSC = min(amountPerSubclass);
%amount of baskets per customer
basketsPerCustomer = zeros(20388,1);
```

```
for j=1:20388
    for i = 1:transactions2
        if indexSortedCustomers2(i) == j
            basketsPerCustomer(j) = basketsPerCustomer(j) + 1;
        end
    end
end
minbasketsPC = min(basketsPerCustomer);
maxbasketsPC = max(basketsPerCustomer);
meanbasketsPC = mean(basketsPerCustomer);
medianbasketsPC = median(basketsPerCustomer);
%aggregated baskets (only necessary for trainingset)
aggregatedBaskets = zeros(20388, 1);

for j = 1:20388
    for i= 1:size(baskets2,1)
        if sortedCustomers2(j) == customerTransaction2(i)
            aggregatedBaskets(j) = aggregatedBaskets(j) + baskets2(i);
        end
    end
end
subclassesInBasket = zeros(20388, 2000);
for i = 1:20388
    i
    newIndexSubClass = find(newCustomerID == sortedCustomers2(i));
    for j = 1:size(newIndexSubClass)
        subclassesInBasket(i,j) = newSubClasses(newIndexSubClass(j));
    end
end
%co occurence matrix
coOccurenceMatrix = zeros(2000, 2000);
for i = 1:20388
    uniqueSubClassesPerBasket = unique(subclassesInBasket(i,:));
    for j = 2:size(uniqueSubClassesPerBasket, 2)-1
        for k = j+1:size(uniqueSubClassesPerBasket,2)
            coOccurenceMatrix(indexMatrix, indexMatrix2) = 1;
        end
    end
end
%generate test set
[sortedCustomersLast, iaLast, indexSortedCustomersLast] = unique(
   customerTransaction2, 'last');
newCustomerIDtest = customerTransaction2(iaLast)';
newIndexNewBasketTest = indexNewBasket(iaLast);
newBasketsTest = baskets2(iaLast)';
newTransDateTest = transDate2(iaLast)';
subclassPerBasketTest = subclassPerBasket(iaLast,:);
newAmountTest = newAmount(iaLast,:);
%remove removed subclasses from baskets
removedSubclasses = uniqueSubClasses(indexZero);
adjustments = 0;
for i = 1:20388
    for j = 1:2000
```

```
        for k = 1:27
            if removedSubclasses(k) == subclassPerBasketTest(i,j)
                subclassPerBasketTest(i,j) = 0;
                adjustments = adjustments + 1;
            end

        end
    end
end
```

Listing 2: Code to generate the training and test set and the similarity matrices

```
indexNewCustomerTrain = [];
for i = 1:20388
    if i == 1
        indexNewCustomerTrain = [indexNewCustomerTrain; i];
    else if sortedCustomerIDtrain(i+1) == sortedCustomerIDtrain(i)
            %do nothing
        else
            indexNewCustomerTrain = [indexNewCustomerTrain; i+1];
        end
    end
end
%determine baskets
userItemMatrix = zeros(20388, 2000);
for i=1:size(indexNewCustomer,1)
    if i == 1
        for j = 1:indexNewCustomer(i+1)-1
            for k = 1:nnz(sortedSubclassPerBasket(indexIDtrain(j),:))
                indexSubClass = find(subclassPerBasketTrain(indexIDtrain(j),k) == uniqueSubClasses);
                userItemMatrix(i,indexSubClass) = userItemMatrix(i,indexSubClass) + size(indexSubClass,2);
            end
        end
    else if i == size(indexNewCustomer,1)
            for j = indexNewCustomer(i):87312
                for k = 1:nnz(sortedSubclassPerBasket(indexIDtrain(j),:))
                    indexSubClass = find(subclassPerBasketTrain(indexIDtrain(j),k) == uniqueSubClasses);
                    userItemMatrix(i,indexSubClass) = userItemMatrix(i,indexSubClass) + size(indexSubClass,2);
                end
            end
        else
            for j = indexNewCustomer(i):indexNewCustomer(i+1) - 1
                for k = 1:nnz(sortedSubclassPerBasket(indexIDtrain(j),:))
                    indexSubClass = find(subclassPerBasketTrain(indexIDtrain(j),k) == uniqueSubClasses);
                    userItemMatrix(i,indexSubClass) = userItemMatrix(i,indexSubClass) + size(indexSubClass,2);
                end
            end
        end
```

```
    end
end
%delete subclasses which are never bought
SUM = sum(userItemMatrix,1);
indexZero = [];
for i=1:2000
    if SUM(i) == 0
        indexZero = [indexZero;i];
    end
end
userItemMatrix2 = userItemMatrix;
userItemMatrix2(:,indexZero)=[];
%delete subclasses which are never bought in the training set
uniqueSubClasses2 = uniqueSubClasses;
uniqueSubClasses2(indexZero) = [];
%define training baskets
trainingBaskets = zeros(20388, 1973);
for i = 1:20388
    for j = 1:1973
        if userItemMatrix2(i,j) > 0
            trainingBaskets(i,j) = uniqueSubClasses2(j);
        end
    end
end
trainingBaskets = sort(trainingBaskets,2, 'descend');
%normalize user-item matrix
logUserItemMatrix = log(userItemMatrix2 + 1);
sumUser = sum(logUserItemMatrix,2);
for i = 1:20388
    for j = 1:1973
        normUserItemMatrix(i,j) = logUserItemMatrix(i,j)/sumUser(i);
    end
end
%cosine based similarity
cosineSim = eye(1973);
for i = 1:1972
    for j=i+1:1973
        cosineSim(i,j) = dot(normUserItemMatrix(:,i), normUserItemMatrix(:,j))/( (
            sqrt(sum(abs(normUserItemMatrix(:,i)).^2))) * (sqrt(sum(abs(
            normUserItemMatrix(:,j)).^2))) );
        cosineSim(j,i) = cosineSim(i,j);
    end
end
%conditional prob. based similarity
for i=1:1973
    freq(i) = nnz(userItemMatrix2(:,i));
end
totAmountOfBoughtProducts = 0;
for i = 1:20388
    totAmountOfBoughtProducts = totAmountOfBoughtProducts + sum(logUserItemMatrix(i
        ,:));
end
for i = 1:1973
```

```
    popularity(i) = sum(logUserItemMatrix(:,i))/totAmountOfBoughtProducts;
end
condProbSimAlpha5 = eye(1973);
condProbSimAlpha7 = eye(1973);
condProbSimAlpha9 = eye(1973);
for i = 1:1972
    for j = i+1:1973
        sumPosEntries = 0;
        for c = 1:20388
            if normUserItemMatrix(c,i)>0
                sumPosEntries = sumPosEntries + normUserItemMatrix(c,j);
            end
        end
        sumPosEntries2 = 0;
        for c = 1:20388
            if normUserItemMatrix(c,j)>0
                sumPosEntries2 = sumPosEntries2 + normUserItemMatrix(c,i);
            end
        end
        condProbSimAlpha5(i,j) = sumPosEntries / (freq(i) * freq(j)^0.5);
        condProbSimAlpha5(j,i) = sumPosEntries2 / (freq(j) * freq(i)^0.5);
        condProbSimAlpha7(i,j) = sumPosEntries / (freq(i) * freq(j)^0.7);
        condProbSimAlpha7(j,i) = sumPosEntries2 / (freq(j) * freq(i)^0.7);
        condProbSimAlpha9(i,j) = sumPosEntries / (freq(i) * freq(j)^0.9);
        condProbSimAlpha9(j,i) = sumPosEntries2 / (freq(j) * freq(i)^0.9);
    end
end
```

Listing 3: Code to determine the transition probability matrix based on a bipartite network for three values of α

```
totProductsPerUser = sum(logUserItemMatrix,2);
totAmountProduct = sum(logUserItemMatrix, 1);
for i = 1:20388
    for j = 1:1973
        probCustToProd5(i,j) = logUserItemMatrix(i,j)/( (totProductsPerUser(i))^0.5 );
        probProdToCust5(i,j) = logUserItemMatrix(i,j)/( (totAmountProduct(j))^0.5 );
        probCustToProd7(i,j) = logUserItemMatrix(i,j)/( (totProductsPerUser(i))^0.7 );
        probProdToCust7(i,j) = logUserItemMatrix(i,j)/( (totAmountProduct(j))^0.7 );
        probCustToProd9(i,j) = logUserItemMatrix(i,j)/( (totProductsPerUser(i))^0.9 );
        probProdToCust9(i,j) = logUserItemMatrix(i,j)/( (totAmountProduct(j))^0.9 );
    end
end
%determine transition probability matrix of bipartite network
bnMatrix5 = zeros(1973, 1973);
bnMatrix7 = zeros(1973, 1973);
bnMatrix9 = zeros(1973, 1973);
for j = 1:1973
    for k=1:1973
        for i = 1:20388
```

```
            bnMatrix5(j,k) = bnMatrix5(j,k) + probCustToProd5(i,j)*probProdToCust5(i,k);
            bnMatrix5(k,j) = bnMatrix5(k,j) + probCustToProd5(i,k)*probProdToCust5(i,j);
            bnMatrix7(j,k) = bnMatrix7(j,k) + probCustToProd7(i,j)*probProdToCust7(i,k);
            bnMatrix7(k,j) = bnMatrix7(k,j) + probCustToProd7(i,k)*probProdToCust7(i,j);
            bnMatrix9(j,k) = bnMatrix9(j,k) + probCustToProd9(i,j)*probProdToCust9(i,k);
            bnMatrix9(k,j) = bnMatrix9(k,j) + probCustToProd9(i,k)*probProdToCust9(i,j);
        end
    end
end
```

Listing 4: Code to determine the ratings for products based on the Random Walk model

```
%determine personalization vector U
Uitem = eye(1973);
%rating CS
RitemCS = inv(eye(1973) - .1*normCS) * (1 - .1) * Uitem;
RitemCS = inv(eye(1973) - .2*normCS) * (1 - .2) * Uitem;
RitemCS = inv(eye(1973) - .3*normCS) * (1 - .3) * Uitem;
RitemCS = inv(eye(1973) - .4*normCS) * (1 - .4) * Uitem;
RitemCS = inv(eye(1973) - .5*normCS) * (1 - .5) * Uitem;
RitemCS = inv(eye(1973) - .6*normCS) * (1 - .6) * Uitem;
RitemCS = inv(eye(1973) - .7*normCS) * (1 - .7) * Uitem;
RitemCS = inv(eye(1973) - .8*normCS) * (1 - .8) * Uitem;
RitemCS = inv(eye(1973) - .9*normCS) * (1 - .9) * Uitem;
%rating CP alpha 0.5
RitemCP5_1 = inv(eye(1973) - .1*normCP5) * (1 - .1) * Uitem;
RitemCP5_2 = inv(eye(1973) - .2*normCP5) * (1 - .2) * Uitem;
RitemCP5_3 = inv(eye(1973) - .3*normCP5) * (1 - .3) * Uitem;
RitemCP5_4 = inv(eye(1973) - .4*normCP5) * (1 - .4) * Uitem;
RitemCP5_5 = inv(eye(1973) - .5*normCP5) * (1 - .5) * Uitem;
RitemCP5_6 = inv(eye(1973) - .6*normCP5) * (1 - .6) * Uitem;
RitemCP5_7 = inv(eye(1973) - .7*normCP5) * (1 - .7) * Uitem;
RitemCP5_8 = inv(eye(1973) - .8*normCP5) * (1 - .8) * Uitem;
RitemCP5_9 = inv(eye(1973) - .9*normCP5) * (1 - .9) * Uitem;
%rating CP alpha 0.7
RitemCP7_1 = inv(eye(1973) - .1*normCP7) * (1 - .1) * Uitem;
RitemCP7_2 = inv(eye(1973) - .2*normCP7) * (1 - .2) * Uitem;
RitemCP7_3 = inv(eye(1973) - .3*normCP7) * (1 - .3) * Uitem;
RitemCP7_4 = inv(eye(1973) - .4*normCP7) * (1 - .4) * Uitem;
RitemCP7_5 = inv(eye(1973) - .5*normCP7) * (1 - .5) * Uitem;
RitemCP7_6 = inv(eye(1973) - .6*normCP7) * (1 - .6) * Uitem;
RitemCP7_7 = inv(eye(1973) - .7*normCP7) * (1 - .7) * Uitem;
RitemCP7_8 = inv(eye(1973) - .8*normCP7) * (1 - .8) * Uitem;
RitemCP7_9 = inv(eye(1973) - .9*normCP7) * (1 - .9) * Uitem;
%rating CP alpha 0.9
RitemCP9_1 = inv(eye(1973) - .1*normCP9) * (1 - .1) * Uitem;
RitemCP9_2 = inv(eye(1973) - .2*normCP9) * (1 - .2) * Uitem;
RitemCP9_3 = inv(eye(1973) - .3*normCP9) * (1 - .3) * Uitem;
```

```
RitemCP9_4 = inv(eye(1973) − .4*normCP9) * (1 − .4) * Uitem;
RitemCP9_5 = inv(eye(1973) − .5*normCP9) * (1 − .5) * Uitem;
RitemCP9_6 = inv(eye(1973) − .6*normCP9) * (1 − .6) * Uitem;
RitemCP9_7 = inv(eye(1973) − .7*normCP9) * (1 − .7) * Uitem;
RitemCP9_8 = inv(eye(1973) − .8*normCP9) * (1 − .8) * Uitem;
RitemCP9_9 = inv(eye(1973) − .9*normCP9) * (1 − .9) * Uitem;
%rating BN alpha 0.5
RitemBN5_1 = inv(eye(1973) − .1*normBnMatrix5) * (1 − .1) * Uitem;
RitemBN5_2 = inv(eye(1973) − .2*normBnMatrix5) * (1 − .2) * Uitem;
RitemBN5_3 = inv(eye(1973) − .3*normBnMatrix5) * (1 − .3) * Uitem;
RitemBN5_4 = inv(eye(1973) − .4*normBnMatrix5) * (1 − .4) * Uitem;
RitemBN5_5 = inv(eye(1973) − .5*normBnMatrix5) * (1 − .5) * Uitem;
RitemBN5_6 = inv(eye(1973) − .6*normBnMatrix5) * (1 − .6) * Uitem;
RitemBN5_7 = inv(eye(1973) − .7*normBnMatrix5) * (1 − .7) * Uitem;
RitemBN5_8 = inv(eye(1973) − .8*normBnMatrix5) * (1 − .8) * Uitem;
RitemBN5_9 = inv(eye(1973) − .9*normBnMatrix5) * (1 − .9) * Uitem;
%rating BN alpha 0.7
RitemBN7_1 = inv(eye(1973) − .1*normBnMatrix7) * (1 − .1) * Uitem;
RitemBN7_2 = inv(eye(1973) − .2*normBnMatrix7) * (1 − .2) * Uitem;
RitemBN7_3 = inv(eye(1973) − .3*normBnMatrix7) * (1 − .3) * Uitem;
RitemBN7_4 = inv(eye(1973) − .4*normBnMatrix7) * (1 − .4) * Uitem;
RitemBN7_5 = inv(eye(1973) − .5*normBnMatrix7) * (1 − .5) * Uitem;
RitemBN7_6 = inv(eye(1973) − .6*normBnMatrix7) * (1 − .6) * Uitem;
RitemBN7_7 = inv(eye(1973) − .7*normBnMatrix7) * (1 − .7) * Uitem;
RitemBN7_8 = inv(eye(1973) − .8*normBnMatrix7) * (1 − .8) * Uitem;
RitemBN7_9 = inv(eye(1973) − .9*normBnMatrix7) * (1 − .9) * Uitem;
%rating BN alpha 0.9
RitemBN9_1 = inv(eye(1973) − .1*normBnMatrix9) * (1 − .1) * Uitem;
RitemBN9_2 = inv(eye(1973) − .2*normBnMatrix9) * (1 − .2) * Uitem;
RitemBN9_3 = inv(eye(1973) − .3*normBnMatrix9) * (1 − .3) * Uitem;
RitemBN9_4 = inv(eye(1973) − .4*normBnMatrix9) * (1 − .4) * Uitem;
RitemBN9_5 = inv(eye(1973) − .5*normBnMatrix9) * (1 − .5) * Uitem;
RitemBN9_6 = inv(eye(1973) − .6*normBnMatrix9) * (1 − .6) * Uitem;
RitemBN9_7 = inv(eye(1973) − .7*normBnMatrix9) * (1 − .7) * Uitem;
RitemBN9_8 = inv(eye(1973) − .8*normBnMatrix9) * (1 − .8) * Uitem;
RitemBN9_9 = inv(eye(1973) − .9*normBnMatrix9) * (1 − .9) * Uitem;
```

Listing 5: Code to determine the ratings of the products based on the evidence products with help of the similarity matrices. The code also contains the computation of the binary hit rate based on a popularity selection of targets. The code for the hit rate based on a random selection of targets is similar to the code based on a popularity selection.

```
%determine targets test basket based on popularity
for i = 1:11813
    for j = 1:nnz(testBaskets(i,:))
        indexSubClassTest = find(testBaskets(i,j) == uniqueSubClasses2);
        freqSubclass(i,j) = popularity(indexSubClassTest);
    end
end
freqSubclass( freqSubclass == 0 ) = Inf;
[sortedFreqSubclass, indexFreqSubclass] = sort(freqSubclass,2,'ascend');
freqSubclass( isinf(freqSubclass) ) = 0;
sortedFreqSubclass( isinf(sortedFreqSubclass) ) = 0;
```

```
for i = 1:11813
    for j = 1:3
        targetSubclass(i,j) = testBaskets(i,indexFreqSubclass(i,j));
    end
    for j = 4:nnz(testBaskets(i,:))
        evidenceSubclass(i,j) = testBaskets(i,indexFreqSubclass(i,j));
    end
end
%determine recommendations
csRows1=zeros(11813,1973);csRows2=zeros(11813,1973);csRows3=zeros(11813,1973);csRows4=zeros(11813,1973);csRows5=zeros(11813,1973);csRows6=zeros(11813,1973);csRows7=zeros(11813,1973);csRows8=zeros(11813,1973);csRows9=zeros(11813,1973);
cp5Rows1=zeros(11813,1973);cp5Rows2=zeros(11813,1973);cp5Rows3=zeros(11813,1973);cp5Rows4=zeros(11813,1973);cp5Rows5=zeros(11813,1973);cp5Rows6=zeros(11813,1973);cp5Rows7=zeros(11813,1973);cp5Rows8=zeros(11813,1973);cp5Rows9=zeros(11813,1973);
cp7Rows1=zeros(11813,1973);cp7Rows2=zeros(11813,1973);cp7Rows3=zeros(11813,1973);cp7Rows4=zeros(11813,1973);cp7Rows5=zeros(11813,1973);cp7Rows6=zeros(11813,1973);cp7Rows7=zeros(11813,1973);cp7Rows8=zeros(11813,1973);cp7Rows9=zeros(11813,1973);
cp9Rows1=zeros(11813,1973);cp9Rows2=zeros(11813,1973);cp9Rows3=zeros(11813,1973);cp9Rows4=zeros(11813,1973);cp9Rows5=zeros(11813,1973);cp9Rows6=zeros(11813,1973);cp9Rows7=zeros(11813,1973);cp9Rows8=zeros(11813,1973);cp9Rows9=zeros(11813,1973);
bn5Rows1=zeros(11813,1973);bn5Rows2=zeros(11813,1973);bn5Rows3=zeros(11813,1973);bn5Rows4=zeros(11813,1973);bn5Rows5=zeros(11813,1973);bn5Rows6=zeros(11813,1973);bn5Rows7=zeros(11813,1973);bn5Rows8=zeros(11813,1973);bn5Rows9=zeros(11813,1973);
bn7Rows1=zeros(11813,1973);bn7Rows2=zeros(11813,1973);bn7Rows3=zeros(11813,1973);bn7Rows4=zeros(11813,1973);bn7Rows5=zeros(11813,1973);bn7Rows6=zeros(11813,1973);bn7Rows7=zeros(11813,1973);bn7Rows8=zeros(11813,1973);bn7Rows9=zeros(11813,1973);
bn9Rows1=zeros(11813,1973);bn9Rows2=zeros(11813,1973);bn9Rows3=zeros(11813,1973);bn9Rows4=zeros(11813,1973);bn9Rows5=zeros(11813,1973);bn9Rows6=zeros(11813,1973);bn9Rows7=zeros(11813,1973);bn9Rows8=zeros(11813,1973);bn9Rows9=zeros(11813,1973);
for i = 1:11813
    for j = 4:nnz(testBaskets(i,:))
        indexSubClassTest = find(testBaskets(i,indexFreqSubclass(i,j)) == uniqueSubClasses2);
        csRows1(i,:) = csRows1(i,:) + RitemCS1(indexSubClassTest,:);
        csRows2(i,:) = csRows2(i,:) + RitemCS2(indexSubClassTest,:);
        csRows3(i,:) = csRows3(i,:) + RitemCS3(indexSubClassTest,:);
        csRows4(i,:) = csRows4(i,:) + RitemCS4(indexSubClassTest,:);
        csRows5(i,:) = csRows5(i,:) + RitemCS5(indexSubClassTest,:);
        csRows6(i,:) = csRows6(i,:) + RitemCS6(indexSubClassTest,:);
        csRows7(i,:) = csRows7(i,:) + RitemCS7(indexSubClassTest,:);
        csRows8(i,:) = csRows8(i,:) + RitemCS8(indexSubClassTest,:);
        csRows9(i,:) = csRows9(i,:) + RitemCS9(indexSubClassTest,:);
        cp5Rows1(i,:) = cp5Rows1(i,:) + RitemCP5_1(indexSubClassTest,:);
        cp5Rows2(i,:) = cp5Rows2(i,:) + RitemCP5_2(indexSubClassTest,:);
        cp5Rows3(i,:) = cp5Rows3(i,:) + RitemCP5_3(indexSubClassTest,:);
        cp5Rows4(i,:) = cp5Rows4(i,:) + RitemCP5_4(indexSubClassTest,:);
```

```
        cp5Rows5(i,:) = cp5Rows5(i,:) + RitemCP5_5(indexSubClassTest,:);
        cp5Rows6(i,:) = cp5Rows6(i,:) + RitemCP5_6(indexSubClassTest,:);
        cp5Rows7(i,:) = cp5Rows7(i,:) + RitemCP5_7(indexSubClassTest,:);
        cp5Rows8(i,:) = cp5Rows8(i,:) + RitemCP5_8(indexSubClassTest,:);
        cp5Rows9(i,:) = cp5Rows9(i,:) + RitemCP5_9(indexSubClassTest,:);
        cp7Rows1(i,:) = cp7Rows1(i,:) + RitemCP7_1(indexSubClassTest,:);
        cp7Rows2(i,:) = cp7Rows2(i,:) + RitemCP7_2(indexSubClassTest,:);
        cp7Rows3(i,:) = cp7Rows3(i,:) + RitemCP7_3(indexSubClassTest,:);
        cp7Rows4(i,:) = cp7Rows4(i,:) + RitemCP7_4(indexSubClassTest,:);
        cp7Rows5(i,:) = cp7Rows5(i,:) + RitemCP7_5(indexSubClassTest,:);
        cp7Rows6(i,:) = cp7Rows6(i,:) + RitemCP7_6(indexSubClassTest,:);
        cp7Rows7(i,:) = cp7Rows7(i,:) + RitemCP7_7(indexSubClassTest,:);
        cp7Rows8(i,:) = cp7Rows8(i,:) + RitemCP7_8(indexSubClassTest,:);
        cp7Rows9(i,:) = cp7Rows9(i,:) + RitemCP7_9(indexSubClassTest,:);
        cp9Rows1(i,:) = cp9Rows1(i,:) + RitemCP9_1(indexSubClassTest,:);
        cp9Rows2(i,:) = cp9Rows2(i,:) + RitemCP9_2(indexSubClassTest,:);
        cp9Rows3(i,:) = cp9Rows3(i,:) + RitemCP9_3(indexSubClassTest,:);
        cp9Rows4(i,:) = cp9Rows4(i,:) + RitemCP9_4(indexSubClassTest,:);
        cp9Rows5(i,:) = cp9Rows5(i,:) + RitemCP9_5(indexSubClassTest,:);
        cp9Rows6(i,:) = cp9Rows6(i,:) + RitemCP9_6(indexSubClassTest,:);
        cp9Rows7(i,:) = cp9Rows7(i,:) + RitemCP9_7(indexSubClassTest,:);
        cp9Rows8(i,:) = cp9Rows8(i,:) + RitemCP9_8(indexSubClassTest,:);
        cp9Rows9(i,:) = cp9Rows9(i,:) + RitemCP9_9(indexSubClassTest,:);
        bn5Rows1(i,:) = bn5Rows1(i,:) + RitemBN5_1(indexSubClassTest,:);
        bn5Rows2(i,:) = bn5Rows2(i,:) + RitemBN5_2(indexSubClassTest,:);
        bn5Rows3(i,:) = bn5Rows3(i,:) + RitemBN5_3(indexSubClassTest,:);
        bn5Rows4(i,:) = bn5Rows4(i,:) + RitemBN5_4(indexSubClassTest,:);
        bn5Rows5(i,:) = bn5Rows5(i,:) + RitemBN5_5(indexSubClassTest,:);
        bn5Rows6(i,:) = bn5Rows6(i,:) + RitemBN5_6(indexSubClassTest,:);
        bn5Rows7(i,:) = bn5Rows7(i,:) + RitemBN5_7(indexSubClassTest,:);
        bn5Rows8(i,:) = bn5Rows8(i,:) + RitemBN5_8(indexSubClassTest,:);
        bn5Rows9(i,:) = bn5Rows9(i,:) + RitemBN5_9(indexSubClassTest,:);
        bn7Rows1(i,:) = bn7Rows1(i,:) + RitemBN7_1(indexSubClassTest,:);
        bn7Rows2(i,:) = bn7Rows2(i,:) + RitemBN7_2(indexSubClassTest,:);
        bn7Rows3(i,:) = bn7Rows3(i,:) + RitemBN7_3(indexSubClassTest,:);
        bn7Rows4(i,:) = bn7Rows4(i,:) + RitemBN7_4(indexSubClassTest,:);
        bn7Rows5(i,:) = bn7Rows5(i,:) + RitemBN7_5(indexSubClassTest,:);
        bn7Rows6(i,:) = bn7Rows6(i,:) + RitemBN7_6(indexSubClassTest,:);
        bn7Rows7(i,:) = bn7Rows7(i,:) + RitemBN7_7(indexSubClassTest,:);
        bn7Rows8(i,:) = bn7Rows8(i,:) + RitemBN7_8(indexSubClassTest,:);
        bn7Rows9(i,:) = bn7Rows9(i,:) + RitemBN7_9(indexSubClassTest,:);
        bn9Rows1(i,:) = bn9Rows1(i,:) + RitemBN9_1(indexSubClassTest,:);
        bn9Rows2(i,:) = bn9Rows2(i,:) + RitemBN9_2(indexSubClassTest,:);
        bn9Rows3(i,:) = bn9Rows3(i,:) + RitemBN9_3(indexSubClassTest,:);
        bn9Rows4(i,:) = bn9Rows4(i,:) + RitemBN9_4(indexSubClassTest,:);
        bn9Rows5(i,:) = bn9Rows5(i,:) + RitemBN9_5(indexSubClassTest,:);
        bn9Rows6(i,:) = bn9Rows6(i,:) + RitemBN9_6(indexSubClassTest,:);
        bn9Rows7(i,:) = bn9Rows7(i,:) + RitemBN9_7(indexSubClassTest,:);
        bn9Rows8(i,:) = bn9Rows8(i,:) + RitemBN9_8(indexSubClassTest,:);
        bn9Rows9(i,:) = bn9Rows9(i,:) + RitemBN9_9(indexSubClassTest,:);
    end
end
%determine place ratings
```

```
[maxCS1, indexMaxCS1] = sort(csRows1, 2, 'descend');
[maxCS2, indexMaxCS2] = sort(csRows2, 2, 'descend');
[maxCS3, indexMaxCS3] = sort(csRows3, 2, 'descend');
[maxCS4, indexMaxCS4] = sort(csRows4, 2, 'descend');
[maxCS5, indexMaxCS5] = sort(csRows5, 2, 'descend');
[maxCS6, indexMaxCS6] = sort(csRows6, 2, 'descend');
[maxCS7, indexMaxCS7] = sort(csRows7, 2, 'descend');
[maxCS8, indexMaxCS8] = sort(csRows8, 2, 'descend');
[maxCS9, indexMaxCS9] = sort(csRows9, 2, 'descend');
[indexCP5_1, indexMaxCP5_1] = sort(cp5Rows1, 2, 'descend');
[indexCP5_2, indexMaxCP5_2] = sort(cp5Rows2, 2, 'descend');
[indexCP5_3, indexMaxCP5_3] = sort(cp5Rows3, 2, 'descend');
[indexCP5_4, indexMaxCP5_4] = sort(cp5Rows4, 2, 'descend');
[indexCP5_5, indexMaxCP5_5] = sort(cp5Rows5, 2, 'descend');
[indexCP5_6, indexMaxCP5_6] = sort(cp5Rows6, 2, 'descend');
[indexCP5_7, indexMaxCP5_7] = sort(cp5Rows7, 2, 'descend');
[indexCP5_8, indexMaxCP5_8] = sort(cp5Rows8, 2, 'descend');
[indexCP5_9, indexMaxCP5_9] = sort(cp5Rows9, 2, 'descend');
[indexCP7_1, indexMaxCP7_1] = sort(cp7Rows1, 2, 'descend');
[indexCP7_2, indexMaxCP7_2] = sort(cp7Rows2, 2, 'descend');
[indexCP7_3, indexMaxCP7_3] = sort(cp7Rows3, 2, 'descend');
[indexCP7_4, indexMaxCP7_4] = sort(cp7Rows4, 2, 'descend');
[indexCP7_5, indexMaxCP7_5] = sort(cp7Rows5, 2, 'descend');
[indexCP7_6, indexMaxCP7_6] = sort(cp7Rows6, 2, 'descend');
[indexCP7_7, indexMaxCP7_7] = sort(cp7Rows7, 2, 'descend');
[indexCP7_8, indexMaxCP7_8] = sort(cp7Rows8, 2, 'descend');
[indexCP7_9, indexMaxCP7_9] = sort(cp7Rows9, 2, 'descend');
[indexCP9_1, indexMaxCP9_1] = sort(cp9Rows1, 2, 'descend');
[indexCP9_2, indexMaxCP9_2] = sort(cp9Rows2, 2, 'descend');
[indexCP9_3, indexMaxCP9_3] = sort(cp9Rows3, 2, 'descend');
[indexCP9_4, indexMaxCP9_4] = sort(cp9Rows4, 2, 'descend');
[indexCP9_5, indexMaxCP9_5] = sort(cp9Rows5, 2, 'descend');
[indexCP9_6, indexMaxCP9_6] = sort(cp9Rows6, 2, 'descend');
[indexCP9_7, indexMaxCP9_7] = sort(cp9Rows7, 2, 'descend');
[indexCP9_8, indexMaxCP9_8] = sort(cp9Rows8, 2, 'descend');
[indexCP9_9, indexMaxCP9_9] = sort(cp9Rows9, 2, 'descend');
[indexBN5_1, indexMaxBN5_1] = sort(bn5Rows1, 2, 'descend');
[indexBN5_2, indexMaxBN5_2] = sort(bn5Rows2, 2, 'descend');
[indexBN5_3, indexMaxBN5_3] = sort(bn5Rows3, 2, 'descend');
[indexBN5_4, indexMaxBN5_4] = sort(bn5Rows4, 2, 'descend');
[indexBN5_5, indexMaxBN5_5] = sort(bn5Rows5, 2, 'descend');
[indexBN5_6, indexMaxBN5_6] = sort(bn5Rows6, 2, 'descend');
[indexBN5_7, indexMaxBN5_7] = sort(bn5Rows7, 2, 'descend');
[indexBN5_8, indexMaxBN5_8] = sort(bn5Rows8, 2, 'descend');
[indexBN5_9, indexMaxBN5_9] = sort(bn5Rows9, 2, 'descend');
[indexBN7_1, indexMaxBN7_1] = sort(bn7Rows1, 2, 'descend');
[indexBN7_2, indexMaxBN7_2] = sort(bn7Rows2, 2, 'descend');
[indexBN7_3, indexMaxBN7_3] = sort(bn7Rows3, 2, 'descend');
[indexBN7_4, indexMaxBN7_4] = sort(bn7Rows4, 2, 'descend');
[indexBN7_5, indexMaxBN7_5] = sort(bn7Rows5, 2, 'descend');
[indexBN7_6, indexMaxBN7_6] = sort(bn7Rows6, 2, 'descend');
[indexBN7_7, indexMaxBN7_7] = sort(bn7Rows7, 2, 'descend');
[indexBN7_8, indexMaxBN7_8] = sort(bn7Rows8, 2, 'descend');
```

```
[indexBN7_9, indexMaxBN7_9] = sort(bn7Rows9, 2, 'descend');
[indexBN9_1, indexMaxBN9_1] = sort(bn9Rows1, 2, 'descend');
[indexBN9_2, indexMaxBN9_2] = sort(bn9Rows2, 2, 'descend');
[indexBN9_3, indexMaxBN9_3] = sort(bn9Rows3, 2, 'descend');
[indexBN9_4, indexMaxBN9_4] = sort(bn9Rows4, 2, 'descend');
[indexBN9_5, indexMaxBN9_5] = sort(bn9Rows5, 2, 'descend');
[indexBN9_6, indexMaxBN9_6] = sort(bn9Rows6, 2, 'descend');
[indexBN9_7, indexMaxBN9_7] = sort(bn9Rows7, 2, 'descend');
[indexBN9_8, indexMaxBN9_8] = sort(bn9Rows8, 2, 'descend');
[indexBN9_9, indexMaxBN9_9] = sort(bn9Rows9, 2, 'descend');
%determine recommendations and hitrates
for i = 1:11813
    a = 0;
    b = 0;
    c = 0;
    d = 0;
    e = 0;
    f = 0;
    g = 0;
    h = 0;
    z = 0;
    for j = 4:1973
        if a < 3
            if ismember(uniqueSubClasses2(indexMaxCS1(i,j-3)),evidenceSubclass(i,:)) > 0
                %do nothing
            else
                a = a+1;
                recCS1(i,a) = uniqueSubClasses2(indexMaxCS1(i,j-3));
            end
        end
        if b < 3
            if ismember(uniqueSubClasses2(indexMaxCS2(i,j-3)),evidenceSubclass(i,:)) > 0
                %do nothing
            else
                b = b+1;
                recCS2(i,b) = uniqueSubClasses2(indexMaxCS2(i,j-3));
            end
        end
        if c < 3
            if ismember(uniqueSubClasses2(indexMaxCS3(i,j-3)),evidenceSubclass(i,:)) > 0
                %do nothing
            else
                c = c+1;
                recCS3(i,c) = uniqueSubClasses2(indexMaxCS3(i,j-3));
            end
        end
        if d < 3
            if ismember(uniqueSubClasses2(indexMaxCS4(i,j-3)),evidenceSubclass(i,:)) > 0
                %do nothing
```

```
            else
                d = d+1;
                recCS4(i,d) = uniqueSubClasses2(indexMaxCS4(i,j-3));
            end
        end
        if e < 3
            if ismember(uniqueSubClasses2(indexMaxCS5(i,j-3)),evidenceSubclass(i,:)) > 0
                %do nothing
            else
                e = e+1;
                recCS5(i,e) = uniqueSubClasses2(indexMaxCS5(i,j-3));
            end
        end
        if f < 3
            if ismember(uniqueSubClasses2(indexMaxCS6(i,j-3)),evidenceSubclass(i,:)) > 0
                %do nothing
            else
                f = f+1;
                recCS6(i,f) = uniqueSubClasses2(indexMaxCS6(i,j-3));
            end
        end
        if g < 3
            if ismember(uniqueSubClasses2(indexMaxCS7(i,j-3)),evidenceSubclass(i,:)) > 0
                %do nothing
            else
                g = g+1;
                recCS7(i,g) = uniqueSubClasses2(indexMaxCS7(i,j-3));
            end
        end
        if h < 3
            if ismember(uniqueSubClasses2(indexMaxCS8(i,j-3)),evidenceSubclass(i,:)) > 0
                %do nothing
            else
                h = h+1;
                recCS8(i,h) = uniqueSubClasses2(indexMaxCS8(i,j-3));
            end
        end
        if z < 3
            if ismember(uniqueSubClasses2(indexMaxCS9(i,j-3)),evidenceSubclass(i,:)) > 0
                %do nothing
            else
                z = z+1;
                recCS9(i,z) = uniqueSubClasses2(indexMaxCS9(i,j-3));
            end
        end
    end
end
%determine hit rate similarities
```

```
bHR_CS1 = zeros(11813,3);bHR_CS2 = zeros(11813,3);bHR_CS3 = zeros(11813,3);bHR_CS4 =
     zeros(11813,3);bHR_CS5 = zeros(11813,3);bHR_CS6 = zeros(11813,3);bHR_CS7 =
    zeros(11813,3);bHR_CS8 = zeros(11813,3);bHR_CS9 = zeros(11813,3);
for i = 1:11813
    for j = 1:3
        for z= 1:3
            if targetSubclass(i,j) == recCS1(i,z)
                bHR_CS1(i,j) = 1;
            end
            if targetSubclass(i,j) == recCS2(i,z)
                bHR_CS2(i,j) = 1;
            end
            if targetSubclass(i,j) == recCS3(i,z)
                bHR_CS3(i,j) = 1;
            end
            if targetSubclass(i,j) == recCS4(i,z)
                bHR_CS4(i,j) = 1;
            end
            if targetSubclass(i,j) == recCS5(i,z)
                bHR_CS5(i,j) = 1;
            end
            if targetSubclass(i,j) == recCS6(i,z)
                bHR_CS6(i,j) = 1;
            end
            if targetSubclass(i,j) == recCS7(i,z)
                bHR_CS7(i,j) = 1;
            end
            if targetSubclass(i,j) == recCS8(i,z)
                bHR_CS8(i,j) = 1;
            end
            if targetSubclass(i,j) == recCS9(i,z)
                bHR_CS9(i,j) = 1;
            end
        end
    end
end
hitCS1=0;hitCS2= 0;hitCS3=0;hitCS4=0;hitCS5=0;hitCS6=0;hitCS7=0;hitCS8=0;hitCS9=0;
for i = 1:11813
    if nnz(bHR_CS1(i,:))>0
        hitCS1 = hitCS1 + 1;
    end
    if nnz(bHR_CS2(i,:))>0
        hitCS2 = hitCS2 + 1;
    end
    if nnz(bHR_CS3(i,:))>0
        hitCS3 = hitCS3 + 1;
    end
    if nnz(bHR_CS4(i,:))>0
        hitCS4 = hitCS4 + 1;
    end
    if nnz(bHR_CS5(i,:))>0
        hitCS5 = hitCS5 + 1;
    end
```

```
    if nnz(bHR_CS6(i,:))>0
        hitCS6 = hitCS6 + 1;
    end
    if nnz(bHR_CS7(i,:))>0
        hitCS7 = hitCS7 + 1;
    end
    if nnz(bHR_CS8(i,:))>0
        hitCS8 = hitCS8 + 1;
    end
    if nnz(bHR_CS9(i,:))>0
        hitCS9 = hitCS9 + 1;
    end
end
percentageHitCS(1) = (hitCS1/11813);
percentageHitCS(2) = (hitCS2/11813);
percentageHitCS(3) = (hitCS3/11813);
percentageHitCS(4) = (hitCS4/11813);
percentageHitCS(5) = (hitCS5/11813);
percentageHitCS(6) = (hitCS6/11813);
percentageHitCS(7) = (hitCS7/11813);
percentageHitCS(8) = (hitCS8/11813);
percentageHitCS(9) = (hitCS9/11813);
```

Listing 6: Code for weighed hit rate based on conditional probability similarity matrix for $\alpha = 0.5$. Same approach can be applied for $\alpha = 0.7$ and $\alpha = 0.9$

```
sumPreviousProb = zeros(11813,1);
sumHitCP5_1 = zeros(11813,1);sumHitCP5_2 = zeros(11813,1);
sumHitCP5_3 = zeros(11813,1);sumHitCP5_4 = zeros(11813,1);
sumHitCP5_5 = zeros(11813,1);sumHitCP5_6 = zeros(11813,1);
sumHitCP5_7 = zeros(11813,1);sumHitCP5_8 = zeros(11813,1);
sumHitCP5_9 = zeros(11813,1);
for i=1:11813
    for j = 1:nnz(testBaskets(i,:))
        %determine target and evidence
        target = testBaskets(i,j);
        evidence = testBaskets(i,:);
        evidence(j) = 0;
        evidence = sort(evidence, 'descend');
        % %determine similarity evidence cosine based
        cp5RowsL10_1 = zeros(1,1973);cp5RowsL10_2 = zeros(1,1973);
        cp5RowsL10_3 = zeros(1,1973);cp5RowsL10_4 = zeros(1,1973);
        cp5RowsL10_5 = zeros(1,1973);cp5RowsL10_6 = zeros(1,1973);
        cp5RowsL10_7 = zeros(1,1973);cp5RowsL10_8 = zeros(1,1973);
        cp5RowsL10_9 = zeros(1,1973);
        for m = 1:nnz(evidence)
            indexSubClassTest = find(evidence(m) == uniqueSubClasses2);
            cp5RowsL10_1(1,:) = cp5RowsL10_1(1,:) + RitemCP5_1(indexSubClassTest,:);
            cp5RowsL10_2(1,:) = cp5RowsL10_2(1,:) + RitemCP5_2(indexSubClassTest,:);
            cp5RowsL10_3(1,:) = cp5RowsL10_3(1,:) + RitemCP5_3(indexSubClassTest,:);
            cp5RowsL10_4(1,:) = cp5RowsL10_4(1,:) + RitemCP5_4(indexSubClassTest,:);
            cp5RowsL10_5(1,:) = cp5RowsL10_5(1,:) + RitemCP5_5(indexSubClassTest,:);
            cp5RowsL10_6(1,:) = cp5RowsL10_6(1,:) + RitemCP5_6(indexSubClassTest,:);
```

```
            cp5RowsL10_7(1,:) = cp5RowsL10_7(1,:) + RitemCP5_7(indexSubClassTest,:);
            cp5RowsL10_8(1,:) = cp5RowsL10_8(1,:) + RitemCP5_8(indexSubClassTest,:);
            cp5RowsL10_9(1,:) = cp5RowsL10_9(1,:) + RitemCP5_9(indexSubClassTest,:);
        end
        [maxCP5_1, indexMaxCP5_L10_1] = sort(cp5RowsL10_1, 2, 'descend');
        [maxCP5_2, indexMaxCP5_L10_2] = sort(cp5RowsL10_2, 2, 'descend');
        [maxCP5_3, indexMaxCP5_L10_3] = sort(cp5RowsL10_3, 2, 'descend');
        [maxCP5_4, indexMaxCP5_L10_4] = sort(cp5RowsL10_4, 2, 'descend');
        [maxCP5_5, indexMaxCP5_L10_5] = sort(cp5RowsL10_5, 2, 'descend');
        [maxCP5_6, indexMaxCP5_L10_6] = sort(cp5RowsL10_6, 2, 'descend');
        [maxCP5_7, indexMaxCP5_L10_7] = sort(cp5RowsL10_7, 2, 'descend');
        [maxCP5_8, indexMaxCP5_L10_8] = sort(cp5RowsL10_8, 2, 'descend');
        [maxCP5_9, indexMaxCP5_L10_9] = sort(cp5RowsL10_9, 2, 'descend');
        hitCP5_L10_1 = 0;hitCP5_L10_2 = 0;hitCP5_L10_3 = 0;hitCP5_L10_4 = 0;
        hitCP5_L10_5 = 0;hitCP5_L10_6 = 0;hitCP5_L10_7 = 0;hitCP5_L10_8 = 0;
        hitCP5_L10_9 = 0;
        %determine recommendations cosine sim
        k = 0;
        a = 0;
        b = 0;
        c = 0;
        d = 0;
        e = 0;
        f = 0;
        g = 0;
        h = 0;
        for z = 1:1973
            if k<1
                if ismember(uniqueSubClasses2(indexMaxCP5_L10_1(1,z)),evidence)>0
                    %do nothing
                else
                    k = k+1;
                    recCP5_L10_1 = uniqueSubClasses2(indexMaxCP5_L10_1(1,z));
                    if target == recCP5_L10_1
                        hitCP5_L10_1= 1;
                    end
                end
            end
        end
        for z = 1:1973
            if a < 1
                if ismember(uniqueSubClasses2(indexMaxCP5_L10_2(1,z)),evidence)>0
                    %do nothing
                else
                    a = a+1;
                    recCP5_L10_2 = uniqueSubClasses2(indexMaxCP5_L10_2(1,z));
                    if target == recCP5_L10_2
                        hitCP5_L10_2 = 1;
                    end
                end
            end
        end
        for z = 1:1973
```

```
            if b < 1
                if ismember(uniqueSubClasses2(indexMaxCP5_L10_3(1,z)),evidence)>0
                    %do nothing
                else
                    b = b+1;
                    recCP5_L10_3 = uniqueSubClasses2(indexMaxCP5_L10_3(1,z));
                    if target == recCP5_L10_3
                        hitCP5_L10_3= 1;
                    end
                end
            end
        end
        for z = 1:1973
            if c < 1
                if ismember(uniqueSubClasses2(indexMaxCP5_L10_4(1,z)),evidence)>0
                    %do nothing
                else
                    c = c+1;
                    recCP5_L10_4 = uniqueSubClasses2(indexMaxCP5_L10_4(1,z));
                    if target == recCP5_L10_4
                        hitCP5_L10_4= 1;
                    end
                end
            end
        end
        for z = 1:1973
            if d < 1
                if ismember(uniqueSubClasses2(indexMaxCP5_L10_5(1,z)),evidence)>0
                    %do nothing
                else
                    d = d+1;
                    recCP5_L10_5 = uniqueSubClasses2(indexMaxCP5_L10_5(1,z));
                    if target == recCP5_L10_5
                        hitCP5_L10_5= 1;
                    end
                end
            end
        end
        for z = 1:1973
            if e < 1
                if ismember(uniqueSubClasses2(indexMaxCP5_L10_6(1,z)),evidence)>0
                    %do nothing
                else
                    e = e+1;
                    recCP5_L10_6 = uniqueSubClasses2(indexMaxCP5_L10_6(1,z));
                    if target == recCP5_L10_6
                        hitCP5_L10_6= 1;
                    end
                end
            end
        end
        for z = 1:1973
            if f < 1
```

```
                if ismember(uniqueSubClasses2(indexMaxCP5_L10_7(1,z)),evidence)>0
                    %do nothing
                else
                    f = f+1;
                    recCP5_L10_7 = uniqueSubClasses2(indexMaxCP5_L10_7(1,z));
                    if target == recCP5_L10_7
                        hitCP5_L10_7= 1;
                    end
                end
            end
        end
        for z = 1:1973
            if g < 1
                if ismember(uniqueSubClasses2(indexMaxCP5_L10_8(1,z)),evidence)>0
                    %do nothing
                else
                    g = g+1;
                    recCP5_L10_8 = uniqueSubClasses2(indexMaxCP5_L10_8(1,z));
                    if target == recCP5_L10_8
                        hitCP5_L10_8= 1;
                    end
                end
            end
        end
        for z = 1:1973
            if h < 1
                if ismember(uniqueSubClasses2(indexMaxCP5_L10_9(1,z)),evidence)>0
                    %do nothing
                else
                    h = h+1;
                    recCP5_L10_9 = uniqueSubClasses2(indexMaxCP5_L10_9(1,z));
                    if target == recCP5_L10_9
                        hitCP5_L10_9= 1;
                    end
                end
            end
        end
        %previous probability target
        indexTarget = find(target == uniqueSubClasses2);
        sumPreviousProb(i) = sumPreviousProb(i) + (1 - popularity(indexTarget));
        sumHitCP5_1(i) = sumHitCP5_1(i) + (1 - popularity(indexTarget))*hitCP5_L10_1
            ;
        sumHitCP5_2(i) = sumHitCP5_2(i) + (1 - popularity(indexTarget))*hitCP5_L10_2
            ;
        sumHitCP5_3(i) = sumHitCP5_3(i) + (1 - popularity(indexTarget))*hitCP5_L10_3
            ;
        sumHitCP5_4(i) = sumHitCP5_4(i) + (1 - popularity(indexTarget))*hitCP5_L10_4
            ;
        sumHitCP5_5(i) = sumHitCP5_5(i) + (1 - popularity(indexTarget))*hitCP5_L10_5
            ;
        sumHitCP5_6(i) = sumHitCP5_6(i) + (1 - popularity(indexTarget))*hitCP5_L10_6
            ;
        sumHitCP5_7(i) = sumHitCP5_7(i) + (1 - popularity(indexTarget))*hitCP5_L10_7
```

```
            ;
        sumHitCP5_8(i) = sumHitCP5_8(i) + (1 - popularity(indexTarget))*hitCP5_L10_8;
        sumHitCP5_9(i) = sumHitCP5_9(i) + (1 - popularity(indexTarget))*hitCP5_L10_9;
    end
    %determine weighted hit rate basket
    wHR_L10_CP5_1(i) = sumHitCP5_1(i)/sumPreviousProb(i);
    wHR_L10_CP5_2(i) = sumHitCP5_2(i)/sumPreviousProb(i);
    wHR_L10_CP5_3(i) = sumHitCP5_3(i)/sumPreviousProb(i);
    wHR_L10_CP5_4(i) = sumHitCP5_4(i)/sumPreviousProb(i);
    wHR_L10_CP5_5(i) = sumHitCP5_5(i)/sumPreviousProb(i);
    wHR_L10_CP5_6(i) = sumHitCP5_6(i)/sumPreviousProb(i);
    wHR_L10_CP5_7(i) = sumHitCP5_7(i)/sumPreviousProb(i);
    wHR_L10_CP5_8(i) = sumHitCP5_8(i)/sumPreviousProb(i);
    wHR_L10_CP5_9(i) = sumHitCP5_9(i)/sumPreviousProb(i);
end
perCP5(1) = sum(wHR_L10_CP5_1)/11813;
perCP5(2) = sum(wHR_L10_CP5_2)/11813;
perCP5(3) = sum(wHR_L10_CP5_3)/11813;
perCP5(4) = sum(wHR_L10_CP5_4)/11813;
perCP5(5) = sum(wHR_L10_CP5_5)/11813;
perCP5(6) = sum(wHR_L10_CP5_6)/11813;
perCP5(7) = sum(wHR_L10_CP5_7)/11813;
perCP5(8) = sum(wHR_L10_CP5_8)/11813;
perCP5(9) = sum(wHR_L10_CP5_9)/11813;
```

Listing 7: Code for macro-averaged hit rate based on the bipartite network transition probability matrix for $\alpha = 0.5$. The same approach can be applied for $\alpha = 0.7$ and $\alpha = 0.9$

```
hitBN5_L10_1 = [];hitBN5_L10_2 = [];hitBN5_L10_3 = [];hitBN5_L10_4 = [];
hitBN5_L10_5 = [];hitBN5_L10_6 = [];hitBN5_L10_7 = [];hitBN5_L10_8 = [];
hitBN5_L10_9 = [];
for i=1:11813
    for j = 1:nnz(testBaskets(i,:))
        %determine target and evidence
        target = testBaskets(i,j);
        evidence = testBaskets(i,:);
        evidence(j) = 0;
        evidence = sort(evidence, 'descend');
        %determine similarity evidence cosine based
        bn5RowsL10_1 = zeros(1,1973);bn5RowsL10_2 = zeros(1,1973);
        bn5RowsL10_3 = zeros(1,1973);bn5RowsL10_4 = zeros(1,1973);
        bn5RowsL10_5 = zeros(1,1973);bn5RowsL10_6 = zeros(1,1973);
        bn5RowsL10_7 = zeros(1,1973);bn5RowsL10_8 = zeros(1,1973);
        bn5RowsL10_9 = zeros(1,1973);
        for m = 1:nnz(evidence)
            indexSubClassTest = find(evidence(m) == uniqueSubClasses2);
            bn5RowsL10_1(1,:) = bn5RowsL10_1(1,:) + RitemBN5_1(indexSubClassTest,:);
            bn5RowsL10_2(1,:) = bn5RowsL10_2(1,:) + RitemBN5_2(indexSubClassTest,:);
            bn5RowsL10_3(1,:) = bn5RowsL10_3(1,:) + RitemBN5_3(indexSubClassTest,:);
            bn5RowsL10_4(1,:) = bn5RowsL10_4(1,:) + RitemBN5_4(indexSubClassTest,:);
            bn5RowsL10_5(1,:) = bn5RowsL10_5(1,:) + RitemBN5_5(indexSubClassTest,:);
```

```
            bn5RowsL10_6(1,:) = bn5RowsL10_6(1,:) + RitemBN5_6(indexSubClassTest,:);
            bn5RowsL10_7(1,:) = bn5RowsL10_7(1,:) + RitemBN5_7(indexSubClassTest,:);
            bn5RowsL10_8(1,:) = bn5RowsL10_8(1,:) + RitemBN5_8(indexSubClassTest,:);
            bn5RowsL10_9(1,:) = bn5RowsL10_9(1,:) + RitemBN5_9(indexSubClassTest,:);
        end
        [maxBN5_1, indexMaxBN5_L10_1] = sort(bn5RowsL10_1, 2, 'descend');
        [maxBN5_2, indexMaxBN5_L10_2] = sort(bn5RowsL10_2, 2, 'descend');
        [maxBN5_3, indexMaxBN5_L10_3] = sort(bn5RowsL10_3, 2, 'descend');
        [maxBN5_4, indexMaxBN5_L10_4] = sort(bn5RowsL10_4, 2, 'descend');
        [maxBN5_5, indexMaxBN5_L10_5] = sort(bn5RowsL10_5, 2, 'descend');
        [maxBN5_6, indexMaxBN5_L10_6] = sort(bn5RowsL10_6, 2, 'descend');
        [maxBN5_7, indexMaxBN5_L10_7] = sort(bn5RowsL10_7, 2, 'descend');
        [maxBN5_8, indexMaxBN5_L10_8] = sort(bn5RowsL10_8, 2, 'descend');
        [maxBN5_9, indexMaxBN5_L10_9] = sort(bn5RowsL10_9, 2, 'descend');
        %determine recommendations cosine sim
        k = 0;
        a = 0;
        b = 0;
        c = 0;
        d = 0;
        e = 0;
        f = 0;
        g = 0;
        h = 0;
        for z = 1:1973
            if k<1
                if ismember(uniqueSubClasses2(indexMaxBN5_L10_1(1,z)),evidence)>0
                    %do nothing
                else
                    k = k+1;
                    recBN5_L10_1 = uniqueSubClasses2(indexMaxBN5_L10_1(1,z));
                    if target == recBN5_L10_1
                        hitBN5_L10_1 = [hitBN5_L10_1; 1];
                    else
                        hitBN5_L10_1 = [hitBN5_L10_1; 0];
                    end
                end
            end
        end
        for z = 1:1973
            if a < 1
                if ismember(uniqueSubClasses2(indexMaxBN5_L10_2(1,z)),evidence)>0
                    %do nothing
                else
                    a = a+1;
                    recBN5_L10_2 = uniqueSubClasses2(indexMaxBN5_L10_2(1,z));
                    if target == recBN5_L10_2
                        hitBN5_L10_2 = [hitBN5_L10_2; 1];
                    else
                        hitBN5_L10_2 = [hitBN5_L10_2; 0];
                    end
                end
            end
```

```
        end
        for z = 1:1973
            if b < 1
                if ismember(uniqueSubClasses2(indexMaxBN5_L10_3(1,z)),evidence)>0
                    %do nothing
                else
                    b = b+1;
                    recBN5_L10_3 = uniqueSubClasses2(indexMaxBN5_L10_3(1,z));
                    if target == recBN5_L10_3
                        hitBN5_L10_3 = [hitBN5_L10_3; 1];
                    else
                        hitBN5_L10_3 = [hitBN5_L10_3; 0];
                    end
                end
             end
        end
        for z = 1:1973
            if c < 1
                if ismember(uniqueSubClasses2(indexMaxBN5_L10_4(1,z)),evidence)>0
                    %do nothing
                else
                    c = c+1;
                    recBN5_L10_4 = uniqueSubClasses2(indexMaxBN5_L10_4(1,z));
                    if target == recBN5_L10_4
                        hitBN5_L10_4 = [hitBN5_L10_4; 1];
                    else
                        hitBN5_L10_4 = [hitBN5_L10_4; 0];
                    end
                end
            end
        end
        for z = 1:1973
            if d < 1
                if ismember(uniqueSubClasses2(indexMaxBN5_L10_5(1,z)),evidence)>0
                    %do nothing
                else
                    d = d+1;
                    recBN5_L10_5 = uniqueSubClasses2(indexMaxBN5_L10_5(1,z));
                    if target == recBN5_L10_5
                        hitBN5_L10_5 = [hitBN5_L10_5; 1];
                    else
                        hitBN5_L10_5 = [hitBN5_L10_5; 0];
                    end
                end
            end
        end
        for z = 1:1973
            if e < 1
                if ismember(uniqueSubClasses2(indexMaxBN5_L10_6(1,z)),evidence)>0
                    %do nothing
                else
                    e = e+1;
                    recBN5_L10_6 = uniqueSubClasses2(indexMaxBN5_L10_6(1,z));
```

```
                if target == recBN5_L10_6
                    hitBN5_L10_6 = [hitBN5_L10_6; 1];
                else
                    hitBN5_L10_6 = [hitBN5_L10_6; 0];
                end
            end
        end
    end
    for z = 1:1973
        if f < 1
            if ismember(uniqueSubClasses2(indexMaxBN5_L10_7(1,z)),evidence)>0
                %do nothing
            else
                f = f+1;
                recBN5_L10_7 = uniqueSubClasses2(indexMaxBN5_L10_7(1,z));
                if target == recBN5_L10_7
                    hitBN5_L10_7 = [hitBN5_L10_7; 1];
                else
                    hitBN5_L10_7 = [hitBN5_L10_7; 0];
                end
            end
        end
    end
    for z = 1:1973
        if g < 1
            if ismember(uniqueSubClasses2(indexMaxBN5_L10_8(1,z)),evidence)>0
                %do nothing
            else
                g = g+1;
                recBN5_L10_8 = uniqueSubClasses2(indexMaxBN5_L10_8(1,z));
                if target == recBN5_L10_8
                    hitBN5_L10_8 = [hitBN5_L10_8; 1];
                else
                    hitBN5_L10_8 = [hitBN5_L10_8; 0];
                end
            end
        end
    end
    for z = 1:1973
        if h < 1
            if ismember(uniqueSubClasses2(indexMaxBN5_L10_9(1,z)),evidence)>0
                %do nothing
            else
                h = h+1;
                recBN5_L10_9 = uniqueSubClasses2(indexMaxBN5_L10_9(1,z));
                if target == recBN5_L10_9
                    hitBN5_L10_9 = [hitBN5_L10_9; 1];
                else
                    hitBN5_L10_9 = [hitBN5_L10_9; 0];
                end
            end
        end
    end
```

```
    end
end
perBN5(1) = sum(hitBN5_L10_1)/size(hitBN5_L10_1,1);
perBN5(2) = sum(hitBN5_L10_2)/size(hitBN5_L10_2,1);
perBN5(3) = sum(hitBN5_L10_3)/size(hitBN5_L10_3,1);
perBN5(4) = sum(hitBN5_L10_4)/size(hitBN5_L10_4,1);
perBN5(5) = sum(hitBN5_L10_5)/size(hitBN5_L10_5,1);
perBN5(6) = sum(hitBN5_L10_6)/size(hitBN5_L10_6,1);
perBN5(7) = sum(hitBN5_L10_7)/size(hitBN5_L10_7,1);
perBN5(8) = sum(hitBN5_L10_8)/size(hitBN5_L10_8,1);
perBN5(9) = sum(hitBN5_L10_9)/size(hitBN5_L10_9,1);
```

Listing 8: Code for user-based Collaborative Filtering model

```
%normalize user user sim matrix
sumUser = sum(userCosineSim, 2);
for i=1:20388
    for j = 1:20388
        normUserCosineSim(i,j) = userCosineSim(i,j)/sumUser(i);
    end
end

%determine mean of the ratings
mean_ratings = zeros(20388,1);
for i = 1:20388
    mean_ratings(i,1) = sum(normUserItemMatrix(i,:))/nnz(normUserItemMatrix(i,:));
end

%determine mean ratings test customers/active user
mean_ratings_test = mean_ratings(indexCustomersTestSet);
tic
%determine k most similar customers
ratings = zeros(11813, 1973);
for i = 1:11813
    indexCustomer = indexCustomersTestSet(i);

    [maxUCS, indexMaxUserCS] = sort(userCosineSim(indexCustomer,:), 'descend');

    for j = 1:1973
        sumRatings = 0;
        for k = 1:3 %nog bepalen tm hoe veel
            sumRatings = sumRatings + normUserCosineSim(indexCustomer,
               indexMaxUserCS(k)) * (normUserItemMatrix(indexMaxUserCS(k),j) -
               mean_ratings(indexMaxUserCS(k)));
        end

        ratings(i,j) = mean_ratings_test(i) + sumRatings;
    end
end
toc
%POPULARITY BASED HIT RATE
%determine targets test basket
for i = 1:11813
    for j = 1:nnz(testBaskets(i,:))
```

```
        indexSubClassTest = find(testBaskets(i,j) == uniqueSubClasses2);
        freqSubclass(i,j) = popularity(indexSubClassTest);
    end
end
freqSubclass( freqSubclass == 0 ) = Inf;
[sortedFreqSubclass, indexFreqSubclass] = sort(freqSubclass,2,'ascend');
freqSubclass( isinf(freqSubclass) ) = 0;
sortedFreqSubclass( isinf(sortedFreqSubclass) ) = 0;
for i = 1:11813
    for j = 1:3
        targetSubclass(i,j) = testBaskets(i,indexFreqSubclass(i,j));
    end
    for j = 4:nnz(testBaskets(i,:))
        evidenceSubclass(i,j) = testBaskets(i,indexFreqSubclass(i,j));
    end
end
%determine recommendations
[maxRating, indexMaxRating] = sort(ratings, 2, 'descend');
for i = 1:11813
    k = 0;
    for j = 4:1973
        if k<3
            if ismember(uniqueSubClasses2(indexMaxRating(i,j-3)),evidenceSubclass(i
                ,:)) > 0
                %do nothing
            else
                k = k+1;
                recExtension_POP(i,k) = uniqueSubClasses2(indexMaxRating(i,j-3));
            end
        end
    end
end
%determine hit rate similarities
binaryHitExtension = zeros(11813,3);

for i = 1:11813
    for j = 1:3
        for k= 1:3
            if targetSubclass(i,j) == recExtension_POP(i,k)
                binaryHitExtension(i,j) = 1;
            end
        end
    end
end
hitPopExtension = 0;
for i = 1:11813
    if nnz(binaryHitExtension(i,:))>0
        hitPopExtension = hitPopExtension + 1;
    end
end
percentageHitPop = hitPopExtension/11813;
%RANDOM BASED HIT RATE
%determine targets test basket
```

```
for i = 1:11813
    randomIndices = randperm(nnz(testBaskets(i,:)),3);
    for j = 1:3
        targetSubclassRandom(i,j) = testBaskets(i,randomIndices(j));
    end
    evidenceSubclassRandom(i,:) = testBaskets(i,:);
    evidenceSubclassRandom(i,randomIndices) = 0;
end
evidenceSubclassRandom = sort(evidenceSubclassRandom,2, 'descend');
%determine recommendations
recExtension_RND = [];
[maxRatingRND, indexMaxRating_RND] = sort(ratings, 2, 'descend');
for i = 1:11813
    k = 0;
    for j = 1:1973
        if k<3
            if ismember(uniqueSubClasses2(indexMaxRating_RND(i,j)),
               evidenceSubclassRandom(i,:)) > 0
                %do nothing
            else
                k = k+1;
                recExtension_RND(i,k) = uniqueSubClasses2(indexMaxRating_RND(i,j));
            end
        end
    end
end
%determine hit rate similarities
binaryHitRandom_EX = zeros(11813,3);

for i = 1:11813
    for j = 1:3
        for k= 1:3
            if targetSubclassRandom(i,j) == recExtension_RND(i,k)
                binaryHitRandom_EX(i,j) = 1;
            end
        end
    end
end
hitRandom_EX = 0;
for i = 1:11813
    if nnz(binaryHitRandom_EX(i,:))>0
        hitRandom_EX = hitRandom_EX + 1;
    end
end
percentageHitRnd = hitRandom_EX/11813;
%WEIGHED HIT RATE
hitExtension = [];
[maxRating_loo, indexMaxRating_loo] = sort(ratings, 2, 'descend');
for i = 1:11813
    for j = 1:nnz(testBaskets(i,:))
        %determine target and evidence
        target = testBaskets(i,j);
        evidence = testBaskets(i,:);
```

```
        evidence(j) = 0;
        evidence = sort(evidence, 'descend');
        %determine recommendation
        k = 0;
        for m = 1:1973
            if k<1
                if ismember(uniqueSubClasses2(indexMaxRating_loo(i,m)),evidence)>0
                    %do nothing
                else
                    k = k+1;
                    recEx_loo = uniqueSubClasses2(indexMaxRating_loo(i,m));
                    if target == recEx_loo
                        hitExtension = [hitExtension; 1];
                    else
                        hitExtension = [hitExtension; 0];
                    end
                end
            end
        end
    end
end
%determine total amount of hits
percentageHitEx = sum(hitExtension)/size(hitExtension,1);
```

Listing 9: Code to generate plots of the performance of the metrics

```
d = [0.1;0.2;0.3;0.4;0.5;0.6;0.7;0.8;0.9];
%plot basket-sensitive random walk based on bipartite network
plot(d,bsrw_bn_5,'-s','Color','magenta','Markersize',10,'LineWidth',2)
hold on
plot(d,bsrw_bn_7,'-x','Color','cyan','Markersize',10,'LineWidth',2)
hold on
plot(d,bsrw_bn_9,'-o','Color','red','Markersize',10,'LineWidth',2)
hold off
grid on
x = xlabel('(1 - d)','fontweight','bold');
x.FontSize = 14;
y = ylabel('bHR(pop)','fontweight','bold');
y.FontSize = 14;
lgd = legend({'\alpha = 0.5','\alpha = 0.7', '\alpha = 0.9'},'Location','northwest');
lgd.FontSize = 15;
%plot macro-averaged hit rate
plot(d,macroBN5,'-s','Color','magenta','Markersize',10,'LineWidth',2)
hold on
plot(d,macroBN7,'-X','Color','cyan','Markersize',10,'LineWidth',2)
hold on
plot(d,macroBN9,'-o','Color','red','Markersize',10,'LineWidth',2)
hold off
grid on
x = xlabel('(1 - d)','fontweight','bold');
x.FontSize = 14;
y = ylabel('macroHR(loo)','fontweight','bold');
y.FontSize = 14;
```

```
lgd = legend({'\alpha = 0.5','\alpha = 0.7', '\alpha = 0.9'},'Location','northwest')
    ;
lgd.FontSize = 15;
```