Recency Adapted Next Basket Recommendation

by

Hanna Hurenkamp

501224

3th of July

Erasmus University Rotterdam
Erasmus School of Economics

Supervisor: Luuk van Maasakkers
Second assessor: Jeffrey Durieux

Abstract

Recommendation methods are in high demand with the rapid development of the e-commerce industry. The better predictions one is able to make, the more pleasant the shopping process for the customer, and the better the expectations for the producers. In this paper, we analyse recommendation methods for next basket predictions and especially the incorporation of the aspect of recency. We try to answer the question on how to include the aspect of time in a general applicable way in next basket recommendation.

We investigate and reproduce the prediction methods as proposed by Faggioli et al. (2020). They use the principle of collaborative filtering (CF) and transform existing methods to recency-aware methods, which only consider a subset of the most recent orders. These recency-aware methods do create more accurate predictions than using the complete order history of an user, but do also discard a substantial part of the available data. Therefore, we propose several time weight methods. To all orders a time weight is assigned; to recent orders a greater time weight is assigned than to orders made long ago. The time weights are based on either a exponential decay function or a logistic decay function. We find that for small recommendation rankings, the time weight methods perform best for the user-based CF method using a exponential decay function. For a complete recommendation of the whole next basket, the recency-aware user-based CF method performs best.

1 Introduction

E-commerce is ever evolving and more importantly increasing in both users and suppliers. Over the past few years supermarkets have joined this industry and are getting a lot of competition from the so called speed delivery grocery services. Therefore, optimising their e-commerce services is of high importance. If one would be able to construct an effective and accurate recommendation system, it would be highly beneficial to both supermarket services and to the customers of these stores. Supermarket services will be able to predict the shopping behaviour of their customers and use this in decision-making processes in for instance advertising or inventory systems. For customers, an accurate recommendation system will make the online shopping experience faster, easier and more pleasant.

In this research we will build on one specific article called recency-aware Collaborative Filtering for Next Basket Recommendation by Faggioli et al. (2020). They propose a recommendation framework that is built on collaborative filtering (CF) and relies mainly on the principles of popularity, similarity and recency. This paper consists partly of a reproduction of this article and therefore will firstly implement their methods and attempt to obtain similar results. The other part of this paper consists of an extension on their research considering the aspect of recency in the recommendation framework. Faggioli et al. (2020) constructed a model in which only a specified number of recent baskets of customers were taken into consideration. This method turned out to make better predictions compared to using the complete shopping history of consumers. However, there are two main issues with this method which we will try to solve. Firstly, a lot of information is discarded since one only considers a certain number of recent baskets. Secondly, this approach selects the same number of recent baskets for every customer; irrespective of the customer's shopping behaviour. The last five baskets for a weekly shopper span a very different time compared to those of a daily shopper.

To improve on these two issues we will focus on the following question: How do we include the aspect of time in a general applicable way in next basket recommendation? This will add to previous research as we will combine the existing recommendation frameworks like collaborative filtering with an optimal use of the available data. We will investigate this by firstly implementing the proposed methods of Faggioli et al. (2020) and then investigating methods which incorporate the aspect of time in making predictions. As we will mainly be focusing on the recency aspect we will aim to make these methods applicable to and accurate for both frequent shoppers and occasional shoppers.

This research therefore provides a model that will be more generally applicable and that uses information more efficiently. Our results show that the implementation of the principle of recency yields better prediction scores for both the recency-aware methods and the time weight methods. The time weight methods do outperform the recency-aware methods for predicting the complete basket of a user, but for predicting the first five or ten items they yield similar results.

The remainder of the paper is structured as follows. Section 2 treats previous researches on this topic and gives an overview of the theory used to construct this paper. Section 3 provides the necessary information on the data, including data characteristics. Section 4 discusses the methodology. Thereafter, Section 5 describes the results of our research. Section 6 states the conclusion of this research. The lessons learned and ideas for future research in Section 7 conclude the paper.

2 Theoretical Framework

Recommendation models have been used for quite some time and are always changing and improving. As part of the great number of possible implementations of machine learning, recommendation models aim to predict ratings a user might give to a specific item and returns a ranking of these prediction for each user. Nowadays, more people make use of the many e-commerce possibilities and therefore models need to be able to process more information and preferably in shorter time. As there are more users of e-commerce services, one also needs to adapt existing models to use the available data in an optimal way to make accurate predictions.

An often used technique in recommender systems is Collaborative Filtering (CF). This technique uses information of all users to construct recommendations for every individual user. The main idea works in two different ways. If other users, similar to the target user, have already purchased a certain item, then it is likely that the target user will appreciate that item as well. Furthermore, if items are similar in some sense, then if the target user has bought one of them, the likelihood that this user will buy the similar item increases (Aiolli, 2013). Using this information Faggioli et al. (2020) follow a standard approach in making a ranking of the items for each user and use this ranking to make predictions of the next move of the user.

The type of predictions that have to be made, depends on the type of user feedback. There are numeric ratings, for instance a movie rating, which are typically composed of triples in the form of (u, i, r); consisting of the rating value r being given by user u to item i. Another form of feedback is called positive-only feedback, which is presented as a pair in the form (u, i), representing a positive interaction between user u and item i, for instance the purchase of an item in a shopping cart. In the case of numeric ratings, one has to make predictions for the missing values in the user-item matrix, which is often treated as a regression task. In the case of positive-only feedback, the user-item matrix consists of of $true$ values for positive user preference, and $false$ values which either denote the dislike for an item or the unfamiliarity with an item. The aim is to predict the $true$ values for each user, which is more related to classification problems Vinagre et al. (2015).

The existing collaborative filtering algorithms can roughly be divided into two main categories: memory-based and model-based CF algorithms. Model-based CF algorithms construct a model of the information contained in the user-item matrix. In this research we focus on memory-based CF algorithms. A priori knowledge about the behaviour of users and items is exploited and used to make a prediction for the target consumer (Aiolli, 2013). In constructing predictions three aspects have a key role; popularity, similarity and recency.

Among others Anderson et al. (2014) argue that the principle of popularity is of great influence in making predictions as it is fairly plausible that if a user consumed an item many times in the past, we could expect it to be likely to be consumed again. This expectation can be supported by means of the power of habits and loyalty. However, another theory is that users are variety-seeking: once they have consumed an item enough times, they prefer exploring new things over exploiting what they already know. As said before CF uses information of all users and based on the individual behaviour of the target user, makes predictions for this specific user. This differs from a more general technique, in which the global popularity of an item is used to make predictions. This global popularity denotes the popularity of the concerned item-based on all users and does not link the target user's individual choice history to its predictions. Faggioli et al. (2020) therefore define the global popularity of an item and the user-wise popularity of an item. The global popularity denotes how often an item occurs in all baskets; the user-wise popularity denotes for each user how often an item occurs in all their baskets.

As explained before, CF can be applied in multiple ways. One can use the similarity of items to make predictions or the similarity among users. The first approach is called item-based CF and investigates the similarity among all products. Sarwar et al. (2001) look into different techniques for computing item-item similarities namely the often used asymmetric cosine similarity, similarity based on correlation and an adjusted cosine similarity. When analysing similar users to predict the target user's behaviour, it is called user-based CF. For this, Faggioli et al. (2020) use a similar approach to determine the similarity by applying the asymmetric cosine similarity.

Faggioli et al. (2020) mainly focus on the incorporation of recency in making next basket predictions. They note that seasonality and drifts in item's popularity are common patterns in grocery shopping such as Christmas products and changes in availability.

The time dimension in CF methods can be approached in different ways. Time-aware models explicitly model time features, such as the day of the month or the hour of the day. Time-dependent models treat ratings as a chronological sequence, implicitly trying to capture temporal dynamics (Shi et al., 2014). For these last models, absolute timestamps in the data are not necessary, since only the chronological order of the data is used.

Faggioli et al. (2020) construct such a time-dependent model and propose a concept of recency-aware user-wise popularity to take these popularity drifts and time-related aspects of this setting into account. They investigate for which recency window, which denotes how many previous baskets of every user are taken into account, they can construct the most accurate predictions. A drawback of this approach is that by applying a recency window over the number of baskets, the assumption is made that the period covered by the most recent r baskets for each user is approximately the same. This assumption is likely to be violated by many users' histories.

Therefore we look at other methods including the aspect of time (Vinagre et al., 2015). As noted by Ding and Li (2005) this previously explained approach can also aggravate sparsity, since a lot of data is discarded. They therefore propose to maintain all data and to include a time weight function and attempt to find appropriate time weights for items that ensures that the items rated recently are able to contribute more to the prediction of the recommendation items. In their proposed algorithm, they choose an exponential form for the time function to achieve the goal, as the exponential time function is widely used in many applications in which it is desirable to gradually decay the history of past behaviour as time goes by (Aggarwal et al., 2004).

Another approach to include the time aspect is performed by Pradel et al. (2011). Where Faggioli et al. (2020) investigates a certain number of previous baskets, Pradel et al. (2011) selects a certain period that has passed with respect to the reference day. This method again has the drawback of discarding possibly useful data.

A different method that does not require exact purchase dates is one proposed in Ding et al. (2006). They propose to include the aspect of recency by comparing the last rating of an item with the first rating. The more this first rating differs from the most previous rating, the less it is included in the construction of the prediction for the next item. This mechanism is used for movie ratings for which ratings are on a scale between zero and five. For binary rating scales the idea of last score of an item differing more and less with the first score will give somewhat less information, since the only outcomes will be zero and one.

Ultimately the aim is to make predictions of a user's next basket. Faggioli et al. (2020) use the similarity and popularity scores to eventually construct the predictions. To evaluate their methods they make use of the normalized Discounted Cumulative Gain (nDCG) (Busa-Fekete et al., 2012). This measure is a widely used evaluation metric for learning-to-rank algorithms and gives in indication on how correctly the predicted ranking is with respect to the true ranking. For numeric feedback another metric is also used, called the Mean Average Error (MAE), which computes the average absolute deviation of recommendations from their true user-specified values (Ding & Li, 2005). As positive-only feedback only has true values of 0 and 1 this deviation from the predicted value gives somewhat less information and therefore the nDCG metric is preferred.

3 Data

We make use of the dataset `Instacart` ("Accessed in 2022. The Instacart Online Grocery Shopping Dataset", 2017), which is also used by Faggioli et al. (2020). It contains information on almost 50,000 products and the aisle and department they are stored in. Besides that the dataset contains information of over one million orders of over 200,000 customers concerning the products they have ordered, the number of days since their prior order and if a certain product has been ordered before. Also information on which day of the week and at which hour of the day their order is made is available. The order in which products are put in one's basket is also known.

3.1 Data preparation and insights

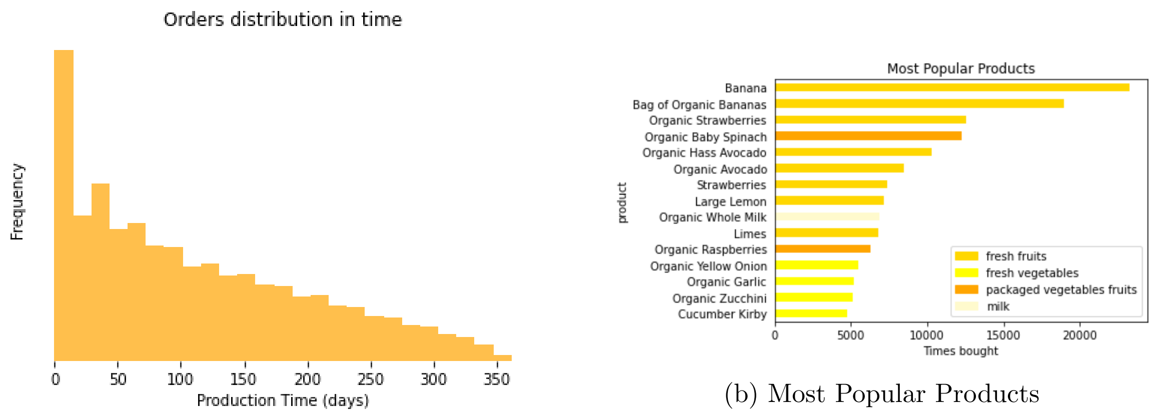
To be able to process the data and to implement our methods on our devices, we take a subset of the data. We choose to include 5% of all users and their most popular 3,000 items. As proposed by Faggioli et al. (2020) we then perform a few pre-processing steps on the dataset. Empty baskets because of the removal of products are discarded and after that users with less than 2 baskets are excluded from further analysis. Table 1 shows the composition of the used subset of the `Instacart` dataset.

Table 1: Dataset composition

Dataset	**Users**	**Items**	**#Baskets**	**min Baskets**	**max Baskets**	**avg #Baskets per User**
`Instacart`	206209	49685	3346083	3	100	16.22
`Subset`	10214	3000	155176	3	100	15.19

Furthermore, we create a new variable called *production time*, which denotes the time from the last purchase to the considered purchase; the number of days that have passed since a 'rating' for an item was produced (Ding & Li, 2005). Since the `Instacart` dataset does not provide exact data, every user has its own timeline. At $t = 0$ the last purchase takes place, and at for the other purchases the value of t depends on their individual *production time*.

Figure 1: Data insights



(a) Orders Distribution in Time

(b) Most Popular Products

We see in Figure 1a that the time span of order history of users is decreasing and that more orders are made recently before the last purchase compared to a year before the purchase. In Figure 1b we see that the most popular products are often found in a few aisles; namely the fresh fruit and vegetables isles, or aisles offering dairy products.

4 Methodology

We will briefly explain the methods of Faggioli et al. (2020), as we reproduce these in the first place. Thereafter, we elaborate on our own proposed methodology.

Let $\mathcal{U}$ be the set of users of cardinality n, and let $\mathcal{I}$ be the set of items of cardinality m. For each user u, we consider an ordered set of transactions b_t^u where t indicates the ordinal position of the grocery shopping, with b_1^u being the first basket and $b_1^{B_u}$ the last for user u. We can define $\mathcal{B}_u = \{b_t^u | t \in 1, ..., B_u\}$ as the set of transactions of a specific user u. We also define the set of baskets of a user u containing a specific item i as $\mathcal{B}_u^i = \{b_t^u | b_t^u \in \mathcal{B}_u \wedge i \in b_t^u\} \subseteq \mathcal{B}_u$, $|\mathcal{B}_u^i| = B_u^i$.

4.1 Baseline Models

Faggioli et al. (2020) use four recommendation models based on the following aspects: global popularity, user-wise popularity, item-based collaborative filtering, user-based collaborative filtering. For the last three models they investigate the improvement when incorporating their proposed method for including the aspect of recency.

4.1.1 Global Popularity (Gpop)

The global popularity of an item, which is, by definition, independent from a single user, is defined as the following:

$$\pi_i = \frac{\sum_{u \in \mathcal{U}} B_u^i}{\sum_{u \in \mathcal{U}} B_u}. \tag{1}$$

This is the most simplistic method in which no information on individual users is used or similarity in any form. Therefore it functions as benchmark model.

4.1.2 User-wise Popularity (UWPop)

The user-wise popularity is the popularity per item computes for each user the user-wise popularity of each item. It is computed by dividing the number of times an item has been bought by the target user by the total number of purchases of this user and is defined as follows:

$$\pi_i^u = \frac{B_u^i}{B_u}. \tag{2}$$

Based on these values the recommendation for the next basket can be composed. Here, the principle of recency-aware user popularity is introduced. Here we only consider the last r baskets of a user in constructing the user-wise popularity. It is defined as follows:

$$\pi_i^u@r = \frac{\sum_{t=[B_u - r]_+}^{B_u} \|i \in b_t^u\|}{\min(r, B_u)}, \tag{3}$$

where r is the size of the recency window, $\|x\|$ is the indicator function which returns 1 if the predicate x is true, 0 otherwise, and $[x]_+$ indicates the maximum between x and 0. Clearly, if $r \geq B_u$ then $\pi_i^u@r = \pi_i^u$. We refer to the recency-aware user popularity as UWPop@r.

4.1.3 Item Popularity-based CF (IB-CF)

The Item Popularity-based CF method aims at recommending baskets that contain popular items similar to the ones of the target user. The notion of similarity is not based on inherent item similarity in this context, but on the reoccurence of certain combinations of products. For instance, a combination of biological soap and biological strawberries; the products themselves inherently differ, but they share a common characteristic. This same idea can be found for other features, like low budget, gluten-free and so on. This information can be used to determine which products are similar to those bought by the user and to recommend these. The similarity between product i and j is based on the number of times an order contains the two products relative to the total amount of orders which contain one of these products. It is constructed in the following way:

$$s(i,j) = \frac{|\mathcal{B}_i \cap \mathcal{B}_j|}{|\mathcal{B}_i|^{1-\alpha}|\mathcal{B}_j|^{\alpha}} = p(i|j)^{\alpha} p(j|i)^{1-\alpha}, \tag{4}$$

where $\mathcal{B}_i$ denotes the set of transactions (i.e., baskets) in which the item i appears and $0 \leq \alpha \leq 1$ is a trade-off parameter which balances the importance of the probabilities. The more often two products are bought together, the higher their similarity score.

This similarity is used in forming the predictions for the next baskets, which are computed in the following way:

$$\hat{r}_i^u = \sum_{j \in \mathcal{I}} s(i,j)^q \pi_j^u, \tag{5}$$

here q denotes the locality as presented by Aiolli (2013). It is an hyper parameter which represents the strength we want to use in enforcing similarity. A higher value of q means that in the neighbourhood of an item we want to keep only highly similar items. For the IB-CF based on the recency-aware user-wise popularity, we refer to IB-CF@r.

4.1.4 User Popularity- based CF (UB-CF)

The User Popularity-based method relies on the similarity among users. It compares users based on their personal set of items they have bought over time. If users have a lot of items in common when comparing their item set, their similarity score will be higher.

$$w(u,v) = \frac{|\mathcal{I}_u \cap \mathcal{I}_v|}{|\mathcal{I}_u|^{\alpha}|\mathcal{I}_v|^{1-\alpha}}, \tag{6}$$

where $\mathcal{I}_u$ denotes the item set of user u and $0 \leq \alpha \leq 1$ represents the same trade-off parameter as previously mentioned in the IB-CF method. Here the similarities are again used to construct the predictions. As it helps to rate new items that can be of interest to the target users, since similar users have appreciated them as well. The predictions are made in the following way:

$$\hat{r}_i^u = \sum_{v \in \mathcal{U}} w(u,v)^q \pi_i^v, \tag{7}$$

where the hyper parameter q serves again as the locality and for which a high value means that we are allowing only users that are really similar to the target one to contribute to the final score. For the UB-CF based on the recency-aware user-wise popularity, we refer to UB-CF@r.

4.2 Time Weight Adapted Models

We will investigate several methods to optimise the recency aspect in making predictions. Since the `Instacart` dataset does not contain specific purchase dates, but only information on the number of days since prior order, we propose an alternative way to still make the predictions time dependent using our constructed variable *production time* as explained in Section 3. We will use the last order in the training set as individual reference date per user.

4.2.1 Time-weight model formulation

The aforementioned recency-aware methods discard a substantial part of data, since not all orders have to be considered for small values of the hyper parameters. Therefore we also apply the time-weight method of Ding and Li (2005). They assume that the time function they use is a monotonic decreasing function, which reduces uniformly with time t and the value of the time weight lies in the range (0,1].

As mentioned in Section 2, Ding and Li (2005) choose an exponential form for the time function based on previous literature. They also discuss the use of a logistic time function.

They include a factor $f(t_{ic})$ in the prediction formula for each item where t_{ic} represents the number of days that have passed since the user's rating was made (*production time*). The time weight function is defined as follows:

$$f(t) = e^{-\lambda t}, \tag{8}$$

where λ is the decay rate and computed by the half-life parameter T_0 in the following way:

$$F(T_0) = (1/2)f(0), \tag{9}$$

$$\lambda = \frac{1}{T_0} \tag{10}$$

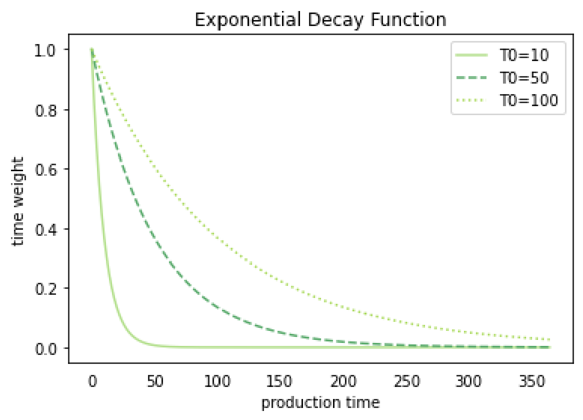
a half-life parameter denotes the time required for a quantity to reduce to half of its initial value. In this case T_0 denotes the time after which half of the weight has been distributed, and therefore $F(T_0)$ can be seen as a cumulative function.

Since no exact purchase dates are available, we will use the last purchase as reference point. We will thus consider a timeline for each user individually. We choose to also consider a logistic time function. The function is defined as follows:

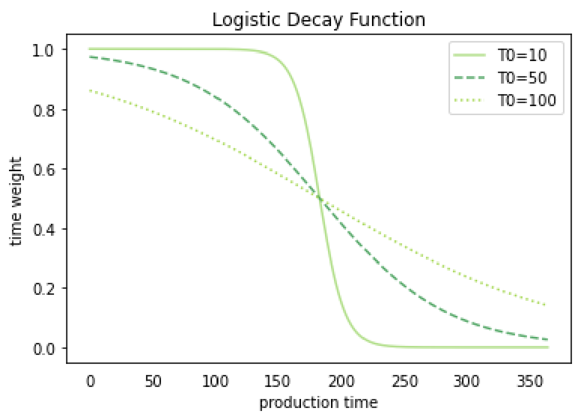
$$f(t) = \frac{1}{1 + e^{\gamma(t - t_0)}} \tag{11}$$

where we use the decay rate γ and t_0 denotes the x value of the inflection point of the function, where the function changes of decreasingly concave down to decreasingly concave up. The larger the value of this inflection point, the longer a large time weight is given to orders. We set this value equal to half a year (183 days), since the data span maximally a year. These functions enable us to assign a time weight to all purchases made and therefore do not discard any data. For both these functions it holds that the value of the time weight reduces with time. The more recent the data, the higher the value of the time weight that is assigned to a purchase.

Figure 2: Decay functions and corresponding time weights



(a) Exponential function for different values of T_0



(b) Logistic function for different values of T_0

In Figures 2a and 2b we observe the difference between the exponential and logistic function and the values they take on for different values of T_0. We see that the exponential function has a steeper curve at the data points that are close to zero compared to data points that are far away from zero. For the logistic function we observe that the gradient of the curve at the middle data point is steepest.

The exponential function thus focuses more on the most recent data, whereas the logistic function has its major decay halfway the total time passed. By investigating both functions we can analyse which time-course applies best to the data. When analysing the influence of the half-time parameter T_0 we take into consideration that it is inversely proportional to decay rates λ and γ; the lower the value of T_0 the higher the decay rate and the faster old data decays and the more focus is laid on the recent data. Optimally we would assign a personal decay rate to each user and to each item group (Ding & Li, 2005). A high value of T_0 would suit customers who are consistent in their preferences for certain items, for instance for dairy products when a consumer buys every week a certain amount of milk and yoghurt. For items such as pie or birthday candles, a lower value of T_0 would be more suitable, since these often are non-recurring purchases. A lower value of T_0 would be more suited for users who frequently change their preferences such that the decay rate will be higher and thus recent data will be given more weight since their order history will not be much help in making prediction due to their changing consuming behaviour.

We will therefore cluster consumers to allow for different values of $T0$. We make groups based on two time related characteristics: the average inter order time and the variance of this inter order time such that consumers who are very consistent in their shopping are grouped together and consumers with a very irregular shopping visits are in another group. As done by Ding and Li (2005) we use simple K-means clustering to divide users into different clusters. The idea of this approach is that it divides the data into clusters such that the distance between the data points per cluster is minimised. To minimise this distance in a fair way, we normalise both the average inter-order time and the variance of this time. We will consider multiple values of T_0 for each cluster in order to find a value that fits the cluster best and optimises the next basket predictions of this cluster.

Similarly to Faggioli et al. (2020) we include this time weight in the user-wise popularity formula; in this way we can implement the time weight aspects in the user-wise popularity predictions and the IB-CF and UB-CF predictions. This time weight aware user-wise popularity function is defined in the following way:

$$\pi_i^u@TW = \frac{\sum_{m=1}^{B_u} \|i \in b_t^u\| f(t_m)}{B_u \sum_{m=1}^{B_u}}, \tag{12}$$

where $\|x\|$ is the indicator function which returns 1 if the predicate x is true, 0 otherwise. Furthermore, m denotes the number of the basket that is considered and t_m the production time of this basket m. $f(t_m)$ denotes the time weight function which either takes on exponential form or logistic form as defined before.

4.3 Train test division and hyper parameter validation

As proposed by Faggioli et al. (2020) in order to build and validate the proposed methods we split the dataset in the following sections: the training set is composed by all baskets but the last one of all users. The validation set is composed by the last baskets of 50% of all users randomly selected and is used to select the best hyper-parameters of the methods. The test set is the set of the remaining 50% of the last baskets.

Since we perform different methods to incorporate the recency aspect, we have different hyper parameters to validate. Firstly, for the recency-aware user popularity of Equation 3 the validation of the hyper parameters has been performed while using the following set of values: for the recency window $r \in \{1, 5, 25, 100, \infty\}$; locality $q \in \{1, 5, 10, 50, 100, 1000\}$. The asymmetry α has a larger set of validation values in main namely $\alpha \in \{0, 0.25, 0.5, 0.75, 1\}$. However, due to limited time we fix α to be equal to 0.5.

For the time weight aware user-wise popularity of Equation 12, we optimise the value of T_0 ϵ {10, 20, 50, 100, 200} based on (Ding & Li, 2005) in order to find the most accurate decay rate λ for the exponential decay function and γ for the logistic decay function. We define the decay rates separately to allow for potential difference. We do this validation process for each cluster separately, to allow for different values of $T0$ for each cluster and thus type of consumer.

4.4 Performance measure

We use the evaluation metric as proposed by Faggioli et al. (2020), which is the Normalised Discounted Cumulative Gain. As explained by Busa-Fekete et al. (2012) the nDCG is used in learning to rank algorithms. The IDCG score denotes the Ideal Discounted Cumulative Gain score, which is the optimal score a ranking is able to obtain. It gives the DCG score if all items in the ranking are ranked correctly. Since we consider the optimal score for a certain test basket, we consider only as many items as can fit into the test basket. To get the nDCG score we investigate how many items are actually ranked correctly and divide this by the optimal possible score (IDCG). The relevance denoted by R_{iu} takes on a value 1 if item i is ranked correctly and takes on 0 if not. Since the relevance score is divided by $\log(i+1)$, the nDCG value increases more for items ranked correctly at higher places in the ranking than for items ranked correctly at the end of the ranking. The formula of the nDCG is defined in Equation 13, in which the second fraction denotes the DCG value.

$$nDCG@k(R_u) = \frac{1}{IDCG@k} \sum_{i=1}^{k} \frac{\mathrm{rel}(R_{iu})}{\log(i+1)}, \tag{13}$$

where

$$IDCG@k = \sum_{i=1}^{\min(k,B)} \frac{1}{\log(i+1)}. \tag{14}$$

As proposed by Faggioli et al. (2020) we fix $k = 5, 10, B$, where B denotes the number of items of the test basket. We observe that for $k < B$, the higher the nDCG score, the more accurate the predictions that have been made. For $k > B$ one has to take into consideration that we predict more items than fit into the test basket, so the chance of getting more items correctly is increased and therefore the nDCG value will be equal or increase for larger k.

5 Results

We firstly show the reproduced results based on the methods of Faggioli et al. (2020). After that, we look at the results obtained from the time weight adapted algorithms for different type of time functions. For all methods we have performed hyper parameter validation beforehand.

5.1 Reproduced results

5.1.1 Hyper parameter validation

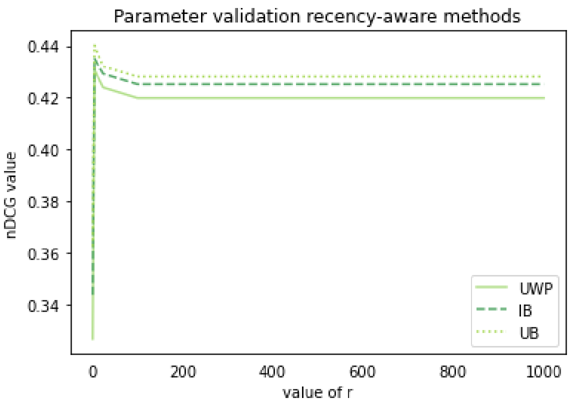
Table 3a shows the optimal hyper parameter values as a result of the hyper parameter validation. Based on this validation process, we opt for a value of $r = 5$ and $q = 5$ as these in general lead to the most accurate predictions. The hyper parameter q represents the strength we want to use to enforce similarity; thus a higher value of q means that only highly similar items are given influence in the predictions. As the value of $q = 5$ is optimal for $q \in \{1, 5, 10, 50, 100, 1000\}$, we see that very strict requirements on similarity do not pay off.

Figure 3b shows the effect on the nDCG scores when selecting different values of the recency window $r \in \{1, 5, 25, 100, \infty\}$. It is clear that a value of $r = 1$ results in the lowest nDCG scores. Besides that the influence of the parameter r is seen most when switching from $r = 1$ to $r = 5$. Hereafter, increasing the value of r does not affect the nDCG value as rapidly as before. Hence, selecting only the five last orders of a user leads to optimal performance of our models.

Figure 3: Hyper parameter validation

(a) Hyper parameter validation outcomes

Method	**r**	**q**	
UWPop@r	5	5	-
	10	5	-
	B	25	-
IB-CF@r	5	5	5
	10	5	5
	B	5	5
UB-CF@r	5	5	5
	10	5	5
	B	5	5



(b) nDCG scores for different values of T_0

5.1.2 Optimal Scores

Table 2 shows the attained nDCG scores, where the bold scores represent the highest scores.

Table 2: nDCG scores of reproduced algorithms

Algorithm	**nDCG@B**	**nDCG@10**	**nDCG@5**
Gpop	0.103	0.108	0.111
UWPop	0.413	0.421	0.417
IB-CF	0.386	0.423	0.421
UB-CF	0.419	0.425	0.421
UWPop@r	0.414	0.427	0.427
IB-CF@r	0.392	0.431	**0.430**
UB-CF@r	**0.425**	**0.433**	**0.430**

We see that the best nDCG values are obtained for the UB-CF@r method. We see also for all methods that correctly prediction the whole basket yields lower scores than for prediction five or ten products. This has to do with the size of the test basket in the models. As explained in Section 4.4 the IDCG@k value depends on the number of items in the test basket. Since half of the test baskets contain less than seven items, the IDCG value for $k = 10$ and $k = B$ is often the same. When considering the DCG factor, however, it will always remain at least the same value when increasing k. Simply put, for larger k the chances increase that the IDCG value remains the same, while the DCG factor stays equal/increases, which leads to higher nDCG scores. The chances of this effect depend on the basket size of the test set and since quite some of our baskets contain less than 10 items, we do have to take this effect into account. This means that we can best look at the value for $k = 5$ and $k = B$, as there are few baskets with less than five items and obviously the aforementioned bias does not present itself for $k = B$.

Besides that, we see that for all methods the recency-aware variants produce better prediction scores. This indicates that we can better use the last five baskets to predict one's next basket, than using all previous baskets.

We also have to note that the obtained scores for the IB-CF and UB-CF method do quite differ from the findings by Faggioli et al. (2020). We have constructed our nDCG scores for the non recent-aware methods already with the optimal hyper parameter $q = 5$. We think that Faggioli et al. (2020) do not do this yet, but keep $q = 1$, since we get similar findings for IB-CF and UB-CF when setting $q = 1$. We choose not to set $q = 1$, since we have already investigated that this does not give optimal results.

5.2 Time weight model results

5.2.1 Hyper parameter Validation and Clustering

For the time weight models, we firstly constructed user clusters based on the average inter order time and the volatility of this time as explained in Section 4. Figure 4 shows the three user clusters that are made, in which the yellow crosses represent the cluster centroids. In our validation set, cluster 0 contains 2075 users, cluster 1 consist of 1101 users and cluster 2 has a size of 1931 users.

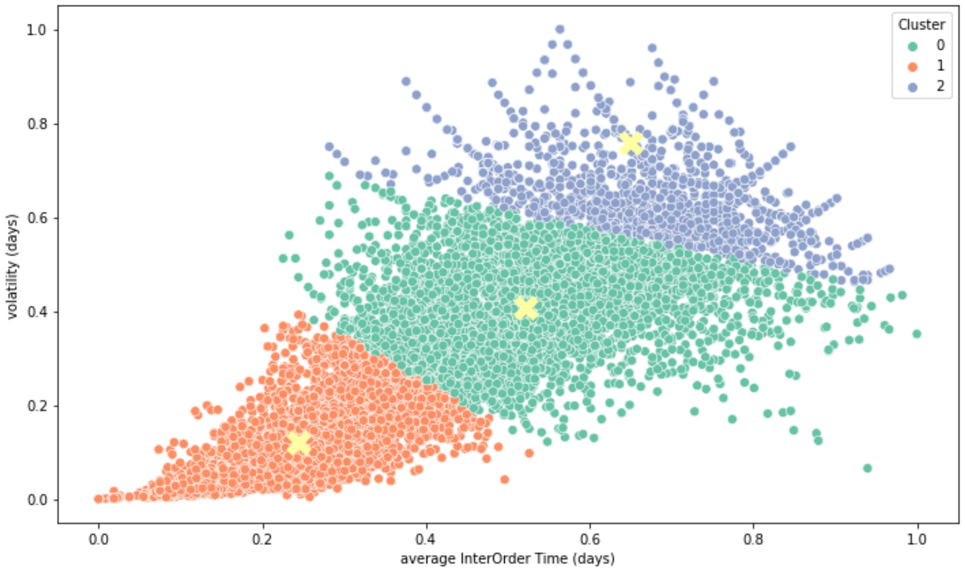


Figure 4: User clusters formed by simple Kmeans method

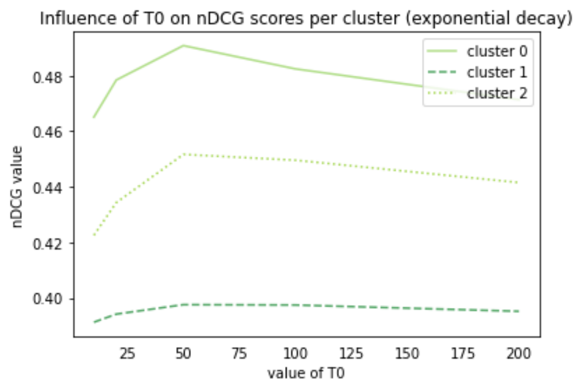
We observe that there is a certain trend in the relation between the average inter order time and the volatility of this time. For small inter order times the volatility is low. Then, when the inter order time increases for the majority of the data the volatility increases as well. There are also quite some points where the volatility decreases again for higher values of the average inter order time. The average inter order time is the time in days between a user's orders on average so this would mean that there is a group of users who consistently (small volatility) do their shoppings very frequently (small inter order time); the group in red with label 1. Part of the turquoise group with label 0 consistently does their shoppings very sparsely. The blue group of users and the majority of the turquoise group is the least consistent in their shoppings as can be seen by the high volatility values.

Table 3 shows the optimal hyper parameter values as a result of the hyper parameter validation. The value for the parameter $(T0_0, T0_1, T0_2)$ denotes the values of the half time parameters for respectively user cluster 0 (green), user cluster 1 (red) and user cluster 2 (blue) for either the exponential decay function or the logistic decay function. Figure 5 shows the effect on the nDCG scores when selecting different values of $T0$.

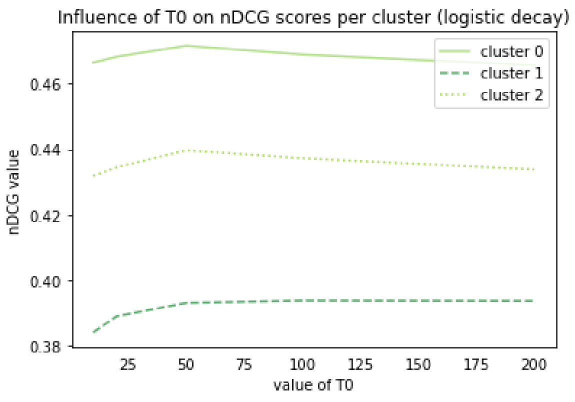
Table 3: Hyper parameter validation outcomes

Method	**k**	**exp**$(T0_1, T0_2, T0_3)$	**log**$(T0_0, T0_1, T0_2)$
UWPop@T0	5	(50, 100, 50)	(50, 100, 50)
	10	(20, 100, 50)	(50, 100, 50)
	B	(50, 100, 50)	(50, 100, 50)
IB-CF@T0	5	(50, 100, 50)	(50, 100, 50)
	10	(50, 100, 50)	(50, 100, 50)
	B	(50, 100, 50)	(50, 100, 50)
UB-CF@T0	5	(50, 50, 50)	(50, 100, 50)
	10	(50, 100, 100)	(50, 100, 50)
	B	(50, 100, 50)	(100, 100, 50)

Figure 5: Influence of $T0$ on nDCG scores per cluster



(a) nDCG scores (UB-CF method) for exponential decay function for different values of T_0



(b) nDCG scores (UB-CF method) for logistic decay function for different values of T_0

Validating on T_0 has shown that for almost all methods and values of k the optimal values of $(T0_0, T0_1, T0_2) = (50, 100, 50)$. The decay rate parameters λ and μ thus have the same value. Furthermore, we notice that cluster 0 and cluster 2 have the same optimal half-time parameter, 50, and thus the same decay rate; $\lambda = \mu = 0.02$. Cluster 1 has a different optimal half-time parameter, 100, which leads to the decay rate $\lambda = \mu = 0.01$. As has been seen in Figure 4 cluster 1 is the group of users with a small average inter order time and a small volatility of this time. A higher decay rate, means that more focus is laid on recent data compared to lower decay rates. A lower decay rate would fit users who are more consistent in their shopping behaviour, which fits the characteristics of the more consistent cluster 1.

Besides that, we observe in Figures 5a and 5b that indeed the course of the nDCG values per clusters changes differently for increasing $T0$. Both for the exponential decay functions and the logistic decay function, we note that cluster 0 and cluster 2 show a similar trend for increasing value of $T0$, containing a slight peak at $T0 = 50$ and decreasing from that point on. For cluster 1 we note that there is no such clear peak in the nDCG values. Moreover, we remark that the predictions per clusters do slightly differ in performance accuracy. For both decay functions cluster 0 obtains the most accurate predictions and cluster 1 the least. A factor of influence could be the cluster size, since we note that in this case the larger the cluster size, the better the predictions. We do have to note, however, that the nDCG score per cluster is averaged over the number of users in the considered cluster, so it cannot be biased for cluster size.

5.2.2 Optimal Results

Table 4 shows the attained nDCG scores where the scores in bold are the highest values. The nCDG scores are averaged over the nDCG scores per cluster, taking into account the size of the clusters. This means that the final nDCG scores are computed as follows:

$$\text{finalNDCG} = \sum_{c=0}^{\text{numClust}} \frac{\text{clustersize}_c}{\text{totUsers}} * \text{nDCG}_c, \tag{15}$$

where clustersize denotes the number of users in the corresponding cluster and totUsers the total number of users in the test set. nDCG_c denotes the highest nDCG score obtained by cluster c. Table 4 shows the final nDCG scores, where the bold scores represent the highest scores.

Table 4: nDCG scores of timeWeight algorithms

TimeFunction	**Algorithm**	**nDCG@B**	**nDCG@10**	**nDCG@5**
exponential	UWPop@T0	0.373	0.434	0.440
	IB-CF@T0	0.383	0.445	0.443
	UB-CF@T0	**0.385**	**0.447**	**0.445**
logistic	UWPop@T0	0.372	0.434	0.430
	IB-CF@T0	0.375	0.436	0.432
	UB-CF@T0	**0.377**	**0.438**	**0.434**

We note that the highest nDCG scores are obtained by using the UB-CF@T0 method. This is similar to the recency-aware results, for which the user-based method also yielded the highest nDCG scores. Furthermore, we observe that the use of the exponential function as decay function results in noticeably better prediction scores compared to using a logistic decay function.

6 Conclusion

In this paper, we try to answer the main question on how to include the aspect of time in a general applicable way in next basket recommendation. We do so by reproducing the methods proposed by Faggioli et al. (2020), which use: global popularity, user-wise popularity, item-based similarity and user-based similarity. They make these methods so called recency-aware by introducing methods which only consider a certain number of the most recent orders. We find, just as they do, that the recency-aware methods perform better than those which use the whole order history without adaptations.

Since their recency-aware methods do discard a lot of information, we investigated time weight methods. These methods do not discard any data and use the principle of time more accurate with regard to the shopping behaviour of both frequent and occasional shoppers. We considered two decay functions; the exponential and logistic decay function. We found that the exponential decay function leads to more accurate predictions. When considering different shapes of these functions, this finding means that it works better to distribute time weight predominantly to recent orders, instead of a larger part of the orders. We find that the time weight method leads to higher nDCG scores for predicting a small set of items. It does, however, not outperform the recency-aware methods when making predictions for the whole basket.

The evaluation metric used is the nDCG metric, which measures how many items are actually ranked correctly, divided by the optimal possible score. As explained before, for this dataset it is most realistic to analyse the accuracy when predicting either the first five items or the complete next basket.

So if one wants to have the most accurate predictions, one should use the exponential time weight method for UB-CF@T0 with varying half-time parameters per user cluster. Then one can predict the top five items per user most correctly, which can be useful for e-commerce if one would like to fill user's baskets automatically with already five products to simplify the shopping process. If one needs predictions to support inventory choices, it would be better to use the time-aware method for UB-CF@r, since it performs best for predicting complete baskets and not only a certain top of products is considered but everything that would need to be in stock.

7 Discussion

As mentioned in the Introduction, a challenge in recommendation systems is the ability to process large amounts of data on users and their choices. In this research, we were only able to use around 5% of the complete dataset, which means that a lot of potentially useful information has not been used. A device that can process more data, or a longer time span to perform research would help, but even then the design of the similarity matrices will always be hard considering memory issues. For further research, a different method which does not require substantial matrices would save a lot of computation time.

Besides using a small subset of the data, we also did not make use of all provided information. There is also information which day of the week or hour of the day an order is placed. One could expect that if someone has been buying milk and bread every Tuesday morning and pancakes every Friday afternoon for several months, that they will do that again next Tuesday morning and Friday afternoon. It could thus be interesting to incorporate the information available on these aspects of the orders.

Lastly, using a time weight function does solve the issue of discarding too much data. One could however optimise this even more by allowing T_0 to be user specific and product group specific. Users have different shopping behaviours and for steady shoppers a lower decay rate would be more suitable, while for inconsistent shoppers a higher decay rate would probably lead to better performances. The same principle holds for product types, e.g. the decay rate for Christmas products can be a lot higher than for milk and bread. Differentiating T_0 for users and product clusters would possibly improve the predictions. Next to differentiating the value of the decay rate, one could also differentiate the inflection point value for the logistic function. In this research, we only applied the logistic function varying the value $T0$. For further research, one could also vary the inflection point value to allow for even more possible shapes of the logistic decay function, which could correspond better to the aspect of recency.

References

Accessed in 2022. The Instacart Online Grocery Shopping Dataset. (2017). https://www.instacart.com/datasets/grocery-shopping-2017

Aggarwal, C. C., Han, J., Wang, J., & Yu, P. S. (2004). A framework for projected clustering of high dimensional data streams. *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, 852–863.

Aiolli, F. (2013). Efficient top-n recommendation for very large scale binary rated datasets. *Proceedings of the 7th ACM conference on Recommender systems*, 273–280.

Anderson, A., Kumar, R., Tomkins, A., & Vassilvitskii, S. (2014). The dynamics of repeat consumption. *Proceedings of the 23rd international conference on World wide web*, 419–430.

Busa-Fekete, R., Szarvas, G., Elteto, T., & Kégl, B. (2012). An apple-to-apple comparison of learning-to-rank algorithms in terms of normalized discounted cumulative gain. *ECAI 2012-20th European Conference on Artificial Intelligence: Preference Learning: Problems and Applications in AI Workshop*, *242*.

Ding, Y., & Li, X. (2005). Time weight collaborative filtering. *Proceedings of the 14th ACM international conference on Information and knowledge management*, 485–492.

Ding, Y., Li, X., & Orlowska, M. E. (2006). Recency-based collaborative filtering. *Proceedings of the 17th Australasian Database Conference-Volume 49*, 99–107.

Faggioli, G., Polato, M., & Aiolli, F. (2020). Recency aware collaborative filtering for next basket recommendation, 80–87. https://doi.org/10.1145/3340631.3394850

Pradel, B., Sean, S., Delporte, J., Guérif, S., Rouveirol, C., Usunier, N., Fogelman-Soulié, F., & Dufau-Joel, F. (2011). A case study in a recommender system based on purchase data. *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 377–385.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. *Proceedings of the 10th international conference on World Wide Web*, 285–295.

Shi, Y., Larson, M., & Hanjalic, A. (2014). Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys (CSUR)*, *47*(1), 1–45.

Vinagre, J., Jorge, A. M., & Gama, J. (2015). An overview on the exploitation of time in collaborative filtering. *Wiley interdisciplinary reviews: Data mining and knowledge discovery*, *5*(5), 195–215.

A Code explanation

In order to obtain the results presented in this thesis, we will give a short guideline to the code.

The code consists of three scripts: onlyDefs, ExtensionA, and parValidation. In the onlyDefs file all reproduced methods are coded and the way in which a subset of the data was taken. The file parValidation contains the codes for the validation of the reproduced results and the ExtensionA file consist of the codes for the methods of my extension and of their validation. For taking a subset of the data, in onlyDefs a method called `SubsetData` can be found, after which the train test validate sets are formed. In this script the methods for creating the predictions for global popularity, user-wise popularity, IB-CF and UB-CF are firstly defined. To obtain the recency-aware variants, we used the recency-aware user-wise popularity computed in `RecencyAwareUWP`.

The validation of the hyperparameters of the reproduced results methods can be found in the file parValidation. Furthermore, all methods needed for our own proposed methods and their validation are stored in the file ExtensionA.