Weighted Neural Collaborative Filtering: Deep Implicit Recommendation with Weighted Positive and Negative Feedback

Stan Hennekes
Erasmus University Rotterdam
Rotterdam, the Netherlands
stanhennekes@gmail.com

Flavius Frasincar
Erasmus University Rotterdam
Rotterdam, the Netherlands
frasincar@ese.eur.nl

ABSTRACT

Being able to generate personalized recommendations is a widespread objective in (online) retail. The focus of this research is to estimate the relevance of user-item combinations based on previous interactions using implicit feedback. We do this in a situation where interactions are often repeated, focusing on new ones. We bring two weighting schemes of positive and negative implicit feedback together into a single Weighted Matrix Factorization (WMF) model to handle the uncertainty associated with implicit preference information. Next, we bring the concept of these weighting schemes to a Deep Learning framework by introducing a Neural Weighted Matrix Factorization model (NeuWMF). We experiment with different weights, loss functions, and regularization terms, and evaluate both models using purchase data from an online supermarket. Our WMF model with both weighted positive and negative feedbacks gives superior performance in terms of NDCG and HR over regular WMF models. Even better results are obtained by our NeuWMF model, which is better capable of capturing the complex patterns behind item preferences. Especially the weighting of positive terms gives an extra boost compared to the state-of-the-art NeuMF model. We confirm the practical use of our model results in an experiment on real customer interactions.

CCS CONCEPTS

• **Information systems → Recommender systems**; *Personalization*; Retrieval tasks and goals.

KEYWORDS

recommender systems, implicit feedback, WMF, NeuMF, NeuWMF

ACM Reference Format:
Stan Hennekes and Flavius Frasincar. 2023. Weighted Neural Collaborative Filtering: Deep Implicit Recommendation with Weighted Positive and Negative Feedback. In *The 38th ACM/SIGAPP Symposium on Applied Computing (SAC '23), March 27-31, 2023, Tallinn, Estonia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3555776.3577619

SAC '23, March 27-31, 2023, Tallinn, Estonia

ACM ISBN 978-1-4503-9517-5/23/03…$15.00
https://doi.org/10.1145/3555776.3577619

1 INTRODUCTION

In online retail, the number of choices can easily overwhelm a customer. To find the relevant items in this overload of options, a user is not always capable or willing to give an explicit query expressing his or her needs and would like to be pointed in the right direction [1]. In everyday life, individuals often rely on recommendations from peers when selecting items. A Recommender System (RS) can help to provide a similar sense of direction for users by automatically generating personalized recommendations [21] based on past user interactions.

In recent years, research effort on RS has shifted towards implicit data [6] [7] [9]. Here, we can think about any interaction a user had with an item, for example purchasing it, viewing it, or clicking on it. If we observe a user interacting with an item, we can interpret this as positive feedback (the user is somehow interested in the item). In contrast, the non-observed user-item pairs are inherently more ambiguous. They partly consist of real negative feedback: the user is not interested in the item. However, some of them can also be regarded as missing values: the user might want to buy the item in the future, may not have seen it at all, or might buy it at a different shop. This uncertainty makes the implicit case challenging.

The main aim of this research is to create a pointwise predicted personalized score of relevance for user-item interactions based on implicit data, where we focus on grocery shopping. Relevant products can be split into two groups, the ones a user did and did not already interact with before. As grocery shopping behavior consists of a large part of repurchases (items a user buys with a certain frequency), we are focused on the latter group. To estimate the relevance of user-item interactions, we propose a combination of different techniques from the field of Collaborative Filtering (CF) that are able to deal with the uncertainty of the context in which a user has made a decision whether or not to interact with an item.

If a user has frequently interacted with an item, we can be more certain about the user's preference for it. If a user interacted with an item once, only to never do it again, it is likely less of their taste than all-time favorites that are interacted with over and over again. We implement this idea in an RS by putting different weights on positive feedback based on the number of interactions [8]. At the same time, we can differentiate in the importance of negative feedback. It is probable that users are more aware of products that are very popular among all users. Therefore, if a user has never interacted with a popular (and assumingly well-known) product, it is likely that this negative feedback was intentional [7]. Again, we can implement the negative feedback in the RS by different weights, this time based on item popularity. We implement both

methods into a single Weighted Matrix Factorization model (WMF) that can be estimated by element-wise Alternating Least Squares. It is already known that implementing either positive or negative feedback is useful in a WMF model, but the combination of the two has not been investigated yet. Next, we bring these concepts of weighted positive and negative feedback to the domain of Deep Learning. We make use of the ability of Deep Learning models to capture highly complex interactions that go further than the only linear relations estimated by WMF. We make use of a Neural Matrix Factorization (NeuMF) framework [6], of which it is known that its performance is superior to WMF models on various domains of implicit datasets, but has not been tested on supermarket data yet. By using the NeuMF framework, we propose a model we call NeuWMF.

To be able to focus on new relevant items, we investigate a modified version of the evaluation measure NCDG next to the standard NDCG and HR. This modified version only rewards the identification of relevant items if they were never interacted with before by the user. We perform an experimental study on historic data from the Dutch online supermarket Picnic and additionally test the results of our model by performing a real-life experiment.

In short, the main contributions of this work are as follows:

(1) We propose NeuWMF: an extension of the NeuMF model that brings the concept of weighting positive and negative feedback to Deep Learning models for implicit recommendation.
(2) We justify the usefulness of weighting positive and negative feedback simultaneously, by studying the combination of both into a single WMF model.
(3) We make two small extensions to be able to focus on the identification of new relevant items in a situation with the repurchase property. Specifically, we define a modified evaluation measure aimed at rewarding new items and a weighting scheme for positive feedback based on relative interaction frequency rather than absolute frequency.
(4) We study the performance of our models and extensions on historic purchase data and in an experimental setting. In both settings, we show that the estimated relevance scores of our best performing model have a significant relation with real conversions.

2 RELATED WORK

Since the 2009 explosion of research effort in the RS field caused by the Netflix prize competition, algorithms like Matrix Factorization (MF) are known to be one of the most effective approaches to Collaborative Filtering [12]. However, many of them focus either on explicit feedback or go beyond purchase information alone [2] [17]. In this work we are focused on model-based CF methods using implicit data. Existing contributions most relevant to our research are summarized in Table 1.

The output of a recommendation system can take multiple forms. The relevance of items can be expressed as the relevance of a single item (pointwise), as a preference of one item over another (pairwise), or as a list of most relevant items to a user (listwise) [21]. Of these three systems, using pointwise relevance is the most versatile approach.

When dealing with implicit recommender systems in a CF framework, several issues are common. Conventional MF techniques often do not work well in the context of implicit feedback, as the interaction matrix is too large and sparse for efficient matrix decomposition. [12] proposed to directly model only the observed interactions. Implicit data is inherently biased and uncertain: we know users interact with items they like, but we do not know why an item is not interacted with. [15] therefore regarded the non-interacted items in implicit data as Missing Not At Random (MNAR) and reformulated the problem of filling the interaction matrix with relevance scores as missing data imputation.

[8] introduced another way of dealing with the uncertain nature of implicit data. They introduced the concept of confidence for positive feedback to distinguish only the most meaningful interactions, leading to a method called Weighted Matrix Factorization (WMF), which we will use in the rest of our research. The weights of specific user-item combinations were taken to be either a linear or a logarithmic function of the number of interactions between them. Estimation of the factors was done by using Alternating Least Squares (ALS). Although the WMF method has proven to give good results, all non-observed user-item combinations are weighted at the same rate, leaving still a lot of uncertainty in this implicit information.

A slightly different approach in capturing the uncertainty of implicit data was taken by [9]. In this work, user and item biases were introduced in a logistic approach of WMF known as Logistic MF. This method even performs well under low numbers of latent factors, leading to improved scalability compared to regular WMF while achieving comparable performance in terms of recall. However, this method was focused on evaluation by Mean Percentage Ranking (MPR) and underperforms for the evaluation methods used in our research (based on our own experiments).

[14] tried to model the implicit uncertainty problem using a latent variable describing whether or not a user has viewed an item (in the case where view data is not directly available). With WMF being a special case of this method, their results are better in terms of precision and recall measures. However, because of the relatively slow Expectation Maximization (EM) algorithm that was used, this method has problems handling large datasets.

More recently, [7] extended the WMF framework to a model they called eALS (after their proposed optimization method) by creating negative feedback weights based on item popularity, also making it possible to distinguish the probability for a user of having seen an item. Since eALS learns latent factors from the whole interaction matrix, it achieves higher accuracy than sampling-based methods. The authors only weighted the negative feedback, regarding positive feedback uniformly.

In the current literature for implicit CF methods, the best performance in terms of precision is obtained by DL models. The introduction of Deep Learning (DL) in the field of RS happened relatively late, with the first DL for RS workshop at ACM RecSys in 2016 [10]. Many attempts used neural networks mainly to model auxiliary information like audio in music recommendation [16] or textual analyses for the recommendation of news articles [4]. More recently, the focus of DL methods has successfully expanded to improve MF methods [6]. In the classical MF approach, latent item and user vectors can only be combined using a dot product,

Table 1: Overview of research contributions for implicit Model-based CF.

Reference	Type	Output	Way of modeling uncertainty
[8]	MF	pointwise	confidence weights
[9]	MF	pointwise	user and item biases
[14]	MF	pointwise	latent viewing variable
[7]	MF	pointwise	confidence weights
[6]	NeuMF	pointwise	-
[5]	NeuMF	pairwise	-

leading to linear interactions. Therefore, classic MF models fail to pick up more complex relationships. The main contribution of deep learning in this context is the fact that it enables to learn non-linear relations as well. [6] proposed the Neural Matrix Factorization (NeuMF) model, a deep neural network model that unifies MF and the Multi-Layer Perceptron into one model for implicit feedback, leading to top-notch and reproducible results [3].

Another successful application of DL was done by [5]. The authors applied a pairwise technique called Bayesian Personalized Ranking (BPR) in a neural network context. BPR [20] uses a different optimization criterion that directly optimizes for pairwise ranking instead of pointwise evaluation and makes use of stochastic gradient descent. Although this method is more efficient, it can not result in pointwise estimates making its application more limited.

While most of the research in Model-based approaches is focused on finding relevant products without regarding timing, other researchers focus on using the order in which items were interacted with as the main source of information [22], [23]. Making use of interaction order information is not in the focus of our research.

3 PRELIMINARIES

We first formalize our problem and discuss existing research related to it. In Section 3.1, we present Matrix Factorization and its extension Weighted Matrix Factorization for implicit feedback. Next, we discuss the Neural Matrix Factorization model in Section 3.2.

3.1 Weighted Matrix Factorization (WMF)

First let us define a user-item interaction matrix $\mathbf{R} \in \mathbb{R}^{M \times N}$ where M denotes the number of users and N the number of items. Each user-item pair (u, i) is represented by the element $r_{ui} = \mathbf{R}(u, i)$ in this matrix. The value of r_{ui} is a measure of the number of interactions user u had with item i (in our context, how often the user purchased the item) and $\mathcal{R}^+$ denotes the set of user-item pairs that are non-zero. The set of user-item pairs without interactions is denoted by $\mathcal{R}^-$, making $\mathcal{R}^+ \cap \mathcal{R}^- = \varnothing$.

The goal of Matrix Factorization (MF) is to decompose the matrix $\mathbf{R}$ into two latent matrices of lower dimensions such that their product approximates $\mathbf{R}$. More precisely, we aim to find two matrices $\mathbf{P} \in \mathbb{R}^{M \times K}$ and $\mathbf{Q} \in \mathbb{R}^{N \times K}$ that are latent factor matrices for users and items. This way, the latent feature space is of dimension K, where K should be much lower than both N and M. Each element r_{ui} can then be estimated as

$$\hat{r}_{ui} = p_u^T q_i, \tag{1}$$

where p_u is the latent feature vector of length K for user u, and q_i the latent feature vector of length K for item i.

If $\mathbf{R}$ were filled with explicit ratings, this model could readily be estimated by minimizing the difference between r_{ui} and $\hat{r}_{ui}$ using a loss function with sufficient regularization. Typically, this is done via gradient descent algorithms. However, in the case of implicit feedback, we have to deal with the uncertainty of the interactions. [8] introduced a weighted regression function where a notion of confidence w_{ui} is defined for each interaction r_{ui} in $\mathcal{R}^+$. [7] extended this WMF framework by creating negative feedback weights c_i for each item. This way, they made it possible to distinguish the probability for a user of having seen an item, but regarded positive feedback uniformly. In this research, we investigate weighting both the positive and negative feedback in a single WMF model.

In order to generate a measure of whether a user is interested in an item or not, the binary variable matrix $\mathbf{B} \in \{0, 1\}^{M \times N}$ is defined, with elements b_{ui} being

$$b_{ui} = \begin{cases} 1, & r_{ui} > 0 \\ 0, & r_{ui} = 0. \end{cases} \tag{2}$$

The model is then estimated by minimizing the following loss function:

$$L = \sum_{(u,i) \in \mathcal{R}^+} w_{ui}(1 - \hat{b}_{ui})^2 + \sum_{(u,i) \in \mathcal{R}^-} c_i \hat{b}_{ui}^2 + \Lambda(p_u, q_i), \tag{3}$$

where $\Lambda(p_u, q_i) = \lambda(\sum_{u=1}^{M} ||p_u||^2 + \sum_{i=1}^{N} ||q_i||^2)$ is a regularization term with regularization constant λ.

3.2 Neural Matrix Factorization (NeuMF)

One of the best performing methods in the field of CF is the Neural Matrix Factorization (NeuMF) model [6]. This method combines the simplicity from the field of MF with the complexity of Neural Network models.

The problem of MF can easily be rewritten in a Neural CF (NCF) framework. If we take the binary user-item interaction matrix of implicit feedback $\mathbf{B}$, the model-based recommendation problem can be abstracted as learning $\hat{b}_{ui} = f(u, i|\Theta)$, where $f(\cdot)$ denotes the function that maps the model parameters Θ to the predicted score $\hat{b}_{ui}$. Note that in the case of MF, the latent factors are combined using a dot product, leading to a linear model of the latent factors.

The limitation to linearity prevents MF methods from picking up complex user-item interactions in the low-dimensional latent space [6], especially in the case of sparse data. This issue can partly be solved by drastically increasing the number of dimensions K, but this severely hurts computation times and may lead to severe cases of overfitting, especially when the interaction matrix is very sparse [18]. This results in a worse generalization of the model. A class of models that is known to be able to pick up these complex interaction patterns are Neural Network models, most noteworthy Multi Layer Perceptron (MLP) models.

On the other hand, we do not want to disregard the MF model completely, as it is known to generate good (and relatively interpretable) results. Although MLP models are more appealing than shallow models in the sense that more complex features can be learned between users and items automatically, shallow models such as CF are particularly well-known to be good at capturing and learning the similarity and implicit relationships between items [24]. Therefore, the idea of NeuMF is to rewrite the MF model to

a neural framework, also known as Generalized MF (GMF), and then combine the results with those of a deep neural network. This network consists of a standard Multi-Layer Perceptron (MLP) that is able to learn the non-linear interactions. A schematic version of the NeuMF model is shown in Figure 1.

The first element of this model thus requires an NCF view on MF. Following the notation of [6] we will call this the GMF, but note that it is just a rewritten MF model. Rewriting an MF to a GMF is quite simple, as we can see MF as a network with a single hidden layer with the following mapping function:

$$\phi^{GMF}(p_u^{GMF}, q_i^{GMF}) = \langle p_u^{GMF}, q_i^{GMF} \rangle \tag{4}$$

The latent vectors p_u and q_i are given the superscript GMF to make clear they are used in the GMF-part of the NeuMF model and we use the notation $\langle a, b \rangle = a^T b$ for the dot product. The output layer can then be constructed as $\hat{b}_{ui} = a_{out}(h^T \phi^{GMF}(p_u^{GMF}, q_i^{GMF}))$ with activation function $a_{out}(\cdot)$ and edge weights h. In the case of simple binary GMF, $a_{out}(\cdot)$ is the identity function and h a vector of ones.

The second element of NeuMF is a standard MLP, where the first layer consists of concatenating the user and item vectors for the interactions: $z_1 = \phi_1^{MLP}(p_u^{MLP}, q_i^{MLP}) = p_u^{MLP}\ q_i^{MLP}$. Subsequently, for the next L layers we have for the x-th layer:

$$\phi_x^{MLP}(z_{x-1}) = a_x(\mathbf{H}_x^T z_{x-1} + \beta_x) \tag{5}$$

with weight matrix $\mathbf{H}_x$, bias vector β_x, and activation function $a_x(\cdot)$ with subscript x for the corresponding layer.

Both elements are combined in the following way, allowing the GMF and MLP parts to learn separate embeddings:

$$\hat{b}_{ui} = \sigma(\gamma^T\ \phi^{GMF}\ \phi_L^{MLP}) \tag{6}$$

where γ determines the weights between the GMF results and the final layer of the MLP.

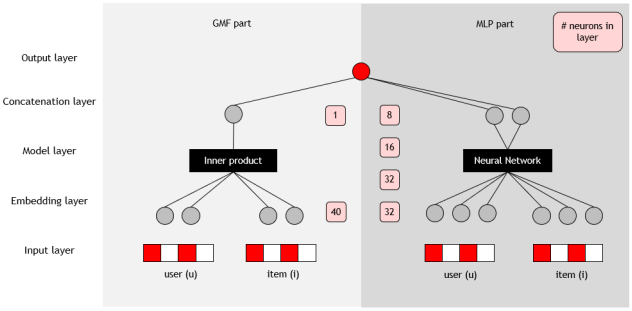


Figure 1: ***Visual representation of the Neu(W)MF model***

For the NeuMF model, the negative loglikelihood is taken as loss function:

$$L_{NeuMF} = -\sum_{(u,i)\in\mathcal{R}} b_{ui} log(\hat{b}_{ui}) + (1 - b_{ui}) log(1 - \hat{b}_{ui}) \tag{7}$$

Estimation then relies on minimizing the negative loglikelihood (by SGD). Note that this leads to solving a binary cross-entropy loss and that this method puts equal weight to all of the user-item interactions.

4 METHODOLOGY

We first discuss the different weighing schemes that can be applied to the WMF model. Next, we integrate weights on the positive and negative feedback into the NeuMF model, leading to a new model that we propose to call NeuWMF.

4.1 WMF with positive and negative weights

To the best of our knowledge, the combination of implementing different levels of confidence in both positive and negative feedback into a WMF model has not been researched yet, although one could expect this to give a significant improvement to the model performance. Previous research efforts are summarized in Table 2.

[8] introduced a weighted regression function where a notion of confidence w_{ui} is defined for each interaction r_{ui} in $\mathcal{R}^+$. This confidence is based on the number of interactions user u had with item i. Before giving the definition of our WMF model we need to make a comment about interactions with high repurchase levels. In this situation, building confidence weights using the number of interactions between u and i might result in getting a bias towards fast-moving products. For example, a basic good like milk (that you probably buy every week) will have a much larger weight than washing liquid (that you only need to buy every once in a while). To remove this bias, we define the confidence weights w_{ui} not on the absolute number of interactions r_{ui} but on the relative frequency of buying an item compared to other users $\tilde{r}_{ui}$. We define $\tilde{r}_{ui} = r_{ui}/\bar{r}_i$, where $\bar{r}_i$ is the average number of purchases of item i of users that ever interacted with it (mean excluding 0). We then use a logarithmic scheme to define the weights:

$$w_{ui} = 1 + \alpha \log(1 + \tilde{r}_{ui}/\epsilon). \tag{8}$$

The reasoning behind the weights w_{ui} is that the more interactions a user has with an item, the more confident we can be that the user actually likes the item. The rate of increase is controlled by the constants $\alpha \in \mathbb{R}^+$ and $\epsilon \in \mathbb{R}_*^+$. The logarithm stems from the idea that this effect decreases for larger amounts of interactions.

While it is clear that all interactions in $\mathcal{R}^+$ can be seen as positive feedback (a user must have at least some level of interest in an item before interacting with it), the combinations in $\mathcal{R}^-$ are more ambiguous. A user may have chosen not to interact with an item, may want to interact with it later, or may not have seen it at all. To model the last of these explanations, [7] proposed to differentiate the weighting of user-item combinations in $\mathcal{R}^-$ based on popularity. The authors argue that all other factors being equal, popular items are more likely to be known by users in general. Therefore, if a user did not interact with a popular item it is more probable that the item is irrelevant than that it is unknown. They formalize this in the parameter c_i, based on item popularity f_i:

$$c_i = c_0 \frac{f_i^\eta}{\sum_{j=1}^N f_j^\eta}. \tag{9}$$

where $f_i = |\mathcal{R}_i^+| / \sum_{j=1}^N |\mathcal{R}_j^+|$ is the fraction of users having interacted with item i compared to the other items. We denote the users that interacted with item i as $\mathcal{R}_i^+$. The parameter $\eta \in \mathbb{R}$ controls the effect of popularity: a value of 1 would lead to a linear relationship between popularity and weight. Setting $\eta \in (0, 1)$ leads to relatively

lower weights c_i for very popular items in the sense that the marginal gain of higher popularity on the weight is damped. Values above 1 lead to an increase in marginal gain. Moreover, $c_0 \in \mathbb{R}^+$ is a scaling parameter that needs to be tuned. The value c_i can be interpreted as the confidence that if i is missed by users, this is true negative feedback.

Table 2: Overview of different WMF weighting schemes in selected literature.

Model	Positive feedback	Negative feedback
WMF_pos [8]	weighted (w_{ui})	uniform ($c_i = 1$)
WMF_neg [7]	uniform ($w_{ui} = 1$)	weighted (c_i)
WMF_pos_neg	weighted (w_{ui})	weighted (c_i)

The objective is to find $\hat{b}_{ui} = p_u^T q_i$, where we use the loss function from Equation 3. Although the optimization of the loss with respect to both p_u and q_i is a complex problem, we can make use of the fact that if either the user factors p_u or the item factors q_i are fixed, the cost function becomes quadratic with respect to the other. Thus, we can derive an analytical expression for the global minimum while holding either the item or user factors fixed and iterate between the two. In order to start this procedure, the matrices **P** and **Q** are randomly initialized. If the negative weights are constant over all items, we can use a computational trick as in [8] and minimize the loss function using Alternating Least Squares (ALS). As we propose to differentiate the negative weights based on item popularity, we need to use the slightly slower element-wise Alternating Least Squares (eALS) method from [7].

4.2 Weighted NeuMF (NeuWMF)

To implement the idea of different levels of confidence in the positive and negative feedback, we propose an extension to the NeuMF model that takes both into account in a similar way as in the WMF model. Therefore, we call this method NeuWMF.

In the input layer of the original NeuMF, every user-item interaction that ever happened is labeled as 1, regardless of how often it occurred. Like in the WMF model from the previous section, we want to include information about how often a certain user bought an item in order to model the uncertainty of item preference when using implicit data. More specifically, we use the number of purchases compared to the number of purchases of the average buyer $\tilde{r}_{ui}$. To do so, we still make use of a variable w_{ui} that is set up like in Equation 8 (so w_{ui} would be equal to 1 for all user-item combinations in $\mathcal{R}^-$). Theoretically, the weight of a user-item interaction can therefore be in the range $[1, \infty)$. In a similar way to the WMF model, we still want to have an output predicting whether or not a user would be interested in an item.

Model estimation can be done in a similar fashion to Equation 3, using a squared loss with a regularization term that takes both the GMF and MLP parameters into account. Note that we chose to use the L2-loss for consistency with the WMF model, whereas the original NeuMF makes use of a binary cross-entropy loss term. In Section 5.2.3, we will show that the effect of this change on the performance of the model is small.

$$L = \sum_{(u,i)\in\mathcal{R}^+} w_{ui}(1-\hat{b}_{ui})^2 + \sum_{(u,i)\in\mathcal{R}^-} c_i \hat{b}_{ui}^2 + \lambda\Big(\sum_{u=1}^{M}(||p_u^{GMF}||^2 + ||p_u^{MLP}||^2) + \sum_{i=1}^{N}(||q_i^{GMF}||^2 + ||q_i^{MLP}||^2)\Big) \tag{10}$$

Due to the sparsity and size of the interaction matrix, it is infeasible to directly calculate the sum over all user-item pairs in $\mathcal{R}^-$. In the original NeuMF, this problem is addressed by uniformly sampling some negative feedback instances in each iteration. That is, for every positive user-item feedback, we sample n negative user-item combinations. It has been empirically shown by [19] that oversampling the most popular - and thus most informative - items can largely help improve convergence without negative effects on performance. In our case, the most informative items are the ones with high negative feedback weights.

In the NeuWMF setting, we can also test the performance when sampling the negative feedback using the weighting from Equation 9. From the values c_i that are obtained by this formula, we sample negative feedback in the following way. The ensemble of item weights is normalized to $\tilde{c}_i$, so $\sum_{i=1}^{N} \tilde{c}_i = 1$. Every negative item is then sampled from a multinomial distribution where the probability to sample item i is equal to $\tilde{c}_i$. So, we approximate the real loss function term $\sum_{(u,i)\in\mathcal{R}^-} c_i \hat{b}_{ui}^2$ with $\sum_{(u,i)\in\mathcal{R}_s^-} \hat{b}_{ui}^2$, where $\mathcal{R}_s^-$ is the sampled version of $\mathcal{R}^-$. Note that $\mathbb{E}(\sum_{(u,i)\in\mathcal{R}_s^-} \hat{b}_{ui}^2) = \sum_{(u,i)\in\mathcal{R}^-}(\mathbb{P}((u,i) \in \mathcal{R}_s^-)\hat{b}_{ui}^2) = \sum_{(u,i)\in\mathcal{R}^-} \tilde{c}_i \hat{b}_{ui}^2$, so we take the negative item weights into account in an implicit manner.

To learn the model parameters, the following loss function is then used:

$$L = \sum_{(u,i)\in\mathcal{R}^+} w_{ui}(1-\hat{b}_{ui})^2 + \sum_{(u,i)\in\mathcal{R}_s^-} \hat{b}_{ui}^2 + \lambda\Big(\sum_{u=1}^{M}(||p_u^{GMF}||^2 + ||p_u^{MLP}||^2) + \sum_{i=1}^{N}(||q_i^{GMF}||^2 + ||q_i^{MLP}||^2)\Big). \tag{11}$$

This loss function can be optimized by performing Stochastic Gradient Descent (SGD). However, to obtain faster convergence we use the Adaptive Moment Estimation (Adam) [11]. This method adapts the learning rate for each parameter on the momentum information from the previous iteration. Therefore, learning rates are more efficiently chosen.

Both the GMF and MLP parts of the NeuWMF model have the same input but have separate and independent embedding layers. We make use of four embedding layers, two for the GMF and two for the MLP part. These layers turn positive integers (indexes) into dense vectors of fixed size, mapping high-dimensional input into a lower-dimensional space such that similar inputs are nearby. For example, the first embedding layer in Figure 1 is the embedding of users in the GMF. The embedding layer is initialized with random values generated from a normal distribution. We use an embedding size of 20 for the GMF and 16 for the MLP. As separate embedding layers are used for items and users in each part, this leads to respectively 40 and 32 neurons in the first layer of the NeuWMF model.

Every user-item combination (including sampled negatives) and the corresponding label are fed to the model. Every user is represented by one-hot encoding into the space of all users (of dimension M) and mapped to a lower-dimensional space, much like the MF model. The number of embeddings in the GMF part is chosen to be close to the number of factors that was used in the WMF model, making the comparison of model performance more natural.

In the GMF part, the embedding layer is followed by a simple multiplication of the resulting vectors (the inner product from WMF). In the MLP part, we make use of several hidden layers with a ReLU activation function. All these layers use a normal initialization of the weights as well. Excluding the embedding layer, we use the optimal MLP set-up from the NeuMF model, using 3 layers with respectively 32, 16, and 8 neurons. The 8 resulting activations from the MLP part are concatenated with the single activation from the GMF part. This is the input for the final layer, which consists of a single output with sigmoid activation function.

As can be seen from Equation 7, the original NeuMF model with binary loss function does not make use of any regularization terms. In the binary case without feedback weights, overfitting might not be too much of a problem in the context of predicting user-item combinations, as every user-item combination can only have one unique observation. However, in the NeuWMF we add extra information on the number of interactions leading to increased risk of overfitting. There are several possible locations to include a regularization term, the most natural one being described in Equation 11. Here, we put the same regularization term λ over all neurons in the embedding layers of the GMF and MLP parts of the model. We briefly experimented with distinguishing separate regularization constants λ^{GMF} and λ^{MLP}, but found no significant advantage and leave a systematic study to further research. Alternatively, we can also add regularization terms in all of the hidden layers of the MLP as well as in the concatenation layer that combines the GMF and MLP outputs. As the input layer of the NeuWMF model contains the most neurons, regularization in this layer can be expected to have the most impact and we leave exploration of the other layers to future research. The model is trained using the batch size of 256 from the NeuMF model. We use no dropouts in order to keep the model structure simple but in the future we would like to experiment with it. In every epoch all positive feedback instances are used, but new negative instances are sampled.

4.3 Evaluation

Like [6], we use Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG) as evaluation measures. The Hit Ratio gives a shallow understanding of success by considering if the interacted item is in a list of recommendations or not, whereas NDCG helps for a better understanding by setting higher scores to hits at higher ranks within such a list. On top of that, we also introduce a slightly modified version of the NDCG that helps us to focus on new user-item interactions by only rewarding relevant combinations if they did not happen before.

The regular HR is the most simple evaluation measure. If we rank our predicted relevance scores per user, we can create a list of recommended items for each user. More formally, the link between the rank rk_{ui} and the relevance b_{ui} for a certain user u is $\text{rk}_{ui} = |\{j : \hat{b}_{uj} \leq \hat{b}_{ui}\}|$. If for a certain user, an item appears in the top k items of a recommended list ($\text{rk}_{ui} \leq k$) and was actually bought during the test period, we call it a hit. Let us define the number of hits appearing in the top k list of user u as $\#\text{hits}_u@k$. The HR for a single user u is then calculated as

$$\text{HR}_u@k = \frac{\#\text{hits}_u@k}{|\text{Test}_u|}, \tag{12}$$

where Test_u is the set of hold-out positive interactions for user u. To get a single measure, the HR is averaged over all users in the test set. The HR is rather intuitive but does not account for the order in which items are recommended. Therefore, we now define the NDCG which will be the main measure throughout this research. To also address the order of top-ranked recommendations, define NDCG for each user as

$$\text{NDCG}_u@k = Z_k \sum_{i=1}^{k} \frac{2^{t_{u,i}} - 1}{log_2(i+1)}, \tag{13}$$

where $t_{u,i}$ represents the real relevance of the ith item on the list of user u. In our case, it is a dummy being 1 if the item is in the users' test set and 0 if not. Note that in this case, the numerator simplifies to $t_{u,i}$. Z_k is a normalizer giving NCDG of 1 for the perfect ranking, being a ranking with only items on top that actually appear in the test set. To get a single measure the NDCG score is then averaged over all users in the test set.

As we are mainly interested in generating recommendations of products a user never bought before, we also introduce a slightly different version of this measure, which we will call the 'NDCG New'. Instead of marking all items bought in the test period as relevant, we only reward those that were bought in the test period for the first time. That is, we train the models on the train data, disregard the resulting scores of all items that user u already interacted with, and rank the rest of the items based on $\hat{b}_{ui}$.

Note that we defined both the HR and NDCG on a list of length k. Typically, the number of unique supermarket items a user interacted with is in the hundreds. However, we usually want a recommended list to be a lot shorter than that. This results in relatively low but more realistic evaluation scores.

5 EXPERIMENTS

We will now describe the experimental setting of our research as well as both the offline and online experiments we performed. Our implementations and an anonymized version of our dataset can be found on GitHub[1].

5.1 Experimental settings

We evaluate the performance of our models on a dataset containing an anonymized version of purchases in the Dutch online supermarket Picnic. Note that because of the fact that we are working with purchase data, all user-item interactions consist of users buying an item.

To reduce the number of data points fed to the model, we use a pre-aggregation of the purchase data. For every user-item combination we take a sum of the number of purchases over time. We do this both during a train period and a test period, where the test

[1] https://github.com/Stan-Hennekes/thesis-weighted-neural-collaborative-filtering

period is later in time. The model is trained on purchase behavior during the train period and evaluated on a test period that follows immediately after. This approach resembles the leave-one-out approach used in other research [6], that artificially splits the latest purchase for each user as a test instance. However, our collective time-split mimics the real-life implementation of an RS better, as in practice we want to be able to make recommendations for all users at the same moment in time. In production, the available information would also be the purchase history of every user until present.

Another reason to choose this method is the repurchase property of grocery data. In the setting of most RS, it is highly unusual that a user interacts many times with a single item (books, movies, news articles). Noteworthy counter-examples are grocery shopping and music recommendation. In the latter contexts, the leave-one-out approach of a single item is not very intuitive; the item that has been left out is likely to be an item the user already interacted with before.

The dataset of the online supermarket was chosen in the following way. The assortment that is available to a customer depends on the region he or she is living in, as the availability of items and capacity of the supply chain is dependent on the local delivery area. Therefore, only users of one specific area at a time are taken into account. We focused on one of the regions where the company has been active for the longest period of time. Purchase data over two years (2017-2018) is taken as train set and we use the following year (2019) as test set. This relatively long test period is taken because of the fact that users tend to stay in their regular purchase patterns and new purchases are therefore relatively rare. Furthermore, by evaluating over a full year we reduce any seasonality effects in purchasing behavior that are known to be common in grocery shopping (most noteworthy the periods before Christmas and Easter).

To make sure that users have enough history to base their preference on, we only consider users that ordered at least 5 times (being the point where they ordered on average about half of the products they ever will). Users should also have at least one order in the test period to base evaluation on and items should be sold in both periods. The top 10 most popular items are left out of the model to prevent them from dominating the latent variables.

As a final check, we assure users and items to have enough positive feedback to build the model on by requiring a user to have bought at least 50 unique items and each item to be bought by at least 50 unique users. Note that these assumptions are relatively mild compared to other research, which often focuses on only the top 5% or even top 1% of most active users. In the end, our analyses are built on an interaction matrix with 5636 users and 4134 items. The sparsity of the train and test matrices are 0.0545 and 0.0379.

5.2 Offline experiments

We will now perform several experiments on real historic purchase data with the models described in the previous sections. In Section 5.2.1, we discuss some model choices specific to our context, such as the tuning of hyperparameters. Thereafter, we turn to an overview of the offline model performance compared to several baselines and explore the differences between some versions of our NeuWMF model in Sections 5.2.2 and 5.2.3. We end this section by a comparison of the most recommended items in the best WMF and NeuWMF models.

5.2.1 Hyperparameter settings. Both the WMF and the NeuWMF models contain several hyperparameters that need to be chosen wisely, most noteworthy the weights we assign to the positive and negative feedback. If we define the expression of w_{ui} and c_i as in Equations 8 and 9, we can tune the hyperparameters α, ϵ, c_0 and η. Furthermore, we have to choose an appropriate regularization constant λ and the number of factors K. Since the required time to train the model is considerable, we took the following approach to find optimal hyperparameters. At first, a random search over all hyperparameters was performed (sampling every hyperparameter uniformly from a specified interval). The hyperparameters and intervals in which searches were performed are shown in Table 3. Subsequently, the best hyperparameters resulting from this search are partially optimized one by one. That is, all hyperparameters are held constant except for one, of which the optimal value is found using a line search over an interval around the optimal value of the grid search. We use the NDCG New as evaluation measure. This optimal value then replaces the previous one in the tuning of the next hyperparameter. Tuning was done on a random subset of 1000 items and 1000 users from the training data to speed up the process.

In this work, we use the length of a recommended list $k = 20$ for all evaluation measures. This value represents the average number of items a user explores on a page in grocery app. From hyperparameter tuning on our experimental dataset, it follows that the effects of ϵ and η are of relatively little importance and we use values of respectively 1 and 0.5. For ϵ this means no special treatment to $\tilde{r}_{ui}$ and for η this is a default value [7].

Next to the ones already introduced in the WMF part, one of the most important choices we need to make for the NeuWMF model is the number of negative samples n. Furthermore, we experimented with different user/item embedding sizes for both the GMF and MLP parts of the model.

Table 3: Ranges of hyperparameters in the WMF model, NeuWMF model and both models used for tuning by grid search and chosen values after partial optimization.

Parameter	Range	Used
K	{1, 2, 3, ..., 200}	20
α^{WMF}	[0, 20]	12
c_0^{WMF}	[0, 100000]	50,000
λ^{WMF}	[0, 10000]	500
α^{NeuWMF}	{0.25, 0.5, 0.75, 1, 1.25, 1.5, 2, 5, 10, 12}	1
c_0^{NeuWMF}	{100, 1000, 10000, 50000}	1000
λ^{NeuWMF}	{[0, 10^{-4}], 1, 10, 100, 500}	0.00005
n	{0, 1, 2, 3, 4, 5}	3
emb. size GMF	{10, 20, 30, 40}	20
emb. size MLP	{8, 16, 24}	16
ϵ	[0.5, 1.5]	1
η	[0, 1]	0.5

5.2.2 Comparison to baselines. Based on the train and test set of historic purchase data, we can compare the performance of our models to some baselines. The resulting evaluation measures are

shown in Table 4. Note that we are only sure that an item was relevant if it was bought during the test period. However, not all relevant items were seen, leading to relatively low evaluation scores for all methods.

Table 4: Results of WMF and NeuWMF models and baselines. Evaluation scores are calculated over the test set.

Model	NDCG	NDCG New	HR
Random	0.03815	0.01972	0.00348
ItemPop	0.35442	0.14820	0.04897
WMF_neg ($w_{ui} = 1$)	0.35335	0.14992	0.04948
WMF_pos ($c_i = 1$)	0.39809	0.15994	0.05002
WMF_pos_neg	0.40989	0.16671	0.05649
NeuMF	0.42499	0.16853	0.06092
NeuWMF	**0.42618**	**0.16998**	**0.06111**

The first baseline consists of a random recommendation, where every user-item combination gets a random relevance score between 0 and 1. The second baseline is a non-personalized approach called ItemPop. This method simply gives the highest relevance scores to the most popular items. Therefore, the top k most popular items amongst all users are given as recommendations for every user u. In the evaluation measure NDCG New, both baselines are evaluated for only the items each user did not buy during training. Therefore, these measures take a form of personalization into account even though both baseline models are non-personalized.

Furthermore, several versions of WMF and NeuMF models are optimized and estimated. The WMF model is estimated as (i) WMF_neg using only negative weights with eALS like [7], (ii) WMF_pos using only positive weights with ALS like [8], and (iii) WMF_pos_neg using both with eALS, as described in Table 2. Every model is evaluated using its optimal values for α and c_0 (if applicable), but the regularization constant λ_{WMF} is constant over the three WMF models. The benefit of combining positive and negative feedback weights in the WMF model is immediately clear. Furthermore, it can be seen that including positive weights has a larger impact on performance than including negative weights.

For the NeuWMF model, regular binary NeuMF is used as a baseline. This vanilla NeuMF does not make use of any regularization. The same number of negative samples is used as for the NeuWMF model. Note that this model already has superior performance over the best WMF model. An extra gain in performance can be achieved by using our weighted NeuWMF model with L2-loss.

5.2.3 Isolating the main components of NeuWMF. To describe the effects of the NeuWMF model in more detail, we conducted additional experiments based on the changes that were made to the NeuMF model to arrive at the NeuWMF model.

Experiment I: Positive feedback. In the NeuMF model, the weights given to every user-item combination are binary. In our NeuWMF model, we make use of the positive weights w_{ui} instead. If we isolate the effect of this change by creating a NeuWMF_pos model (using the same naming conventions as for the WMF), we get the results shown in Table 5. No regularization is used for a fair comparison and therefore only regular NDCG and HR are shown. It can be seen that using weighted positive feedback has a positive impact on all evaluation measures compared to using binary weights. This effect is the main reason that NeuWMF outperforms NeuMF.

Table 5: NeuMF model with L2-loss versus the NeuWMF model with positive weights and uniform sampling of negatives. No regularization is used.

Model	NDCG test	HR test	NDCG train	HR train
NeuMF_L2	0.42275	0.06086	0.58519	0.05931
NeuWMF_pos	**0.42566**	**0.06125**	**0.58933**	**0.05971**

Experiment 2: Negative feedback. In the original NeuMF model, the negative user-item instances are sampled uniformly. In our NeuWMF model, we make use of the negative weights c_i as the probability to be sampled. If we isolate this effect in a NeuWMF_neg model, we get the results shown in Table 6. It can be seen that sampling using negative weights improves performance on the train set, but regularization will be needed on the test set.

Table 6: NeuMF model with L2-loss versus NeuWMF model with negative feedback sampling (but no positive weights). No regularization is used.

Model	NDCG test	HR test	NDCG train	HR train
NeuMF_L2	**0.42275**	**0.06086**	0.58519	0.05931
NeuWMF_neg	0.41174	0.05963	**0.60190**	**0.06179**

Experiment 3: L2-loss. As mentioned in Section 3, the NeuMF model makes use of a binary cross-entropy loss function for estimation. To enable natural comparison to the WMF model, we chose to use an L2-loss function. In Table 7 the performance of the model under both loss functions is shown. It can be seen that the differences in performance are small, with a slight preference towards the L2-loss when regarding new items only. This validates our hypothesis that changing the loss function of the NeuMF from cross-entropy loss to L2-loss does not hurt performance too much.

Table 7: Result of the NeuMF model using L2-loss (NeuWMF without any weights) versus vanilla NeuMF model with cross-entropy as the loss function. No regularization is used.

Model	NDCG	NDCG New
NeuMF_L2	0.42275	**0.16997**
NeuMF_cross-entropy	**0.42499**	0.16853

Experiment 4: Regularization. The NeuMF model without feedback weights does not use any regularization terms. In the binary case without feedback weights, overfitting might not be too much of a problem in the context of predicting user-item combinations, as every user-item combination can only have one unique observation. However, in the NeuWMF we add extra information on the number of interactions leading to increased risk of overfitting. This effect is shown in Table 8. By using regularization in the embedding layer of the NeuWMF, the performance on the test set improves significantly in terms of both normal NDCG and NDCG New. This leads to the best performing model in our context, as we focus on NCDG New in the test set.

Table 8: Result of NeuWMF model using L2-loss with versus without regularization in the embedding layer.

Model	NDCG train	NDCG test	NDCG new test
NeuWMF_no_reg	**0.57242**	0.40117	0.14736
NeuWMF_with_reg	0.49353	**0.42618**	**0.16998**

5.2.4 Interpretation of most recommended items. As we are evaluating based on the highest-ranked items per user, it is of interest to study the items that are ranked top most often. As can be seen in Table 9, the items that are generally ranked high over all users are quite generic. This can of course be expected, as a personalized model averaged over all users again boils down to the most popular products. Also, note that the amount of users having the same first-ranked product is lower for the NeuWMF model, indicating that NeuWMF is better than WMF in picking up personalized preferences other than the most popular items.

Table 9: Most first-ranked articles for WMF_pos_neg and NeuWMF model

WMF_pos_neg	# users	NeuWMF	# users
Seedless white grapes	842	Minced beef	708
Bin bags	729	Bin bags	409
Mushrooms	647	Seedless white grapes	344
Garlic	627	Garlic	297
Avocado	526	Avocado	283
Washed spinach	297	Tangerines	255
Minced beef	279	Onions	233
Onions	217	Medium eggs	228
Tangerines	184	Mushrooms	200
Medium eggs	173	Cauliflower	171

5.3 Online experiments

Apart from the theoretical performance of the RS that was shown in the previous section, it is also of high importance to test the practical usability of the predicted relevance scores. We created a page based on our NeuWMF model that was shown to a group of users selected in the same way as the Picnic set described in Section 5.1. In order to test the page, some business logic had to be applied, for example to meet profitability standards. Specifically, the items were selected based on a score representing the usefulness of including item i in the test page defined as $usefulness_i = margin_i * (Max_u(\hat{b}_{ui}) - Min_u(\hat{b}_{ui})) * Mean_u(\hat{b}_{ui}) * NB_i$, where NB_i is the fraction of users that has never bought product i before. From this list the top available products were selected, resulting in a page of 40 different items.

We showed this page to 4,269 users, of which 3,665 were active in the weeks of the test. 1,413 of these users actually opened the page and had a look at it. 461 of these users bought at least one item from the page. We can now compare the conversion of user-item combinations on this page to how relevant they were predicted to be by the model.

If we order the predicted relevance scores of the 4,134 items that we included in our NeuWMF model for each user, we can give each user-item combination a predicted relevance rank. In doing so, we leave out the items a user already interacted with, as we are interested in finding new relevant combinations. We would expect items that users chose to buy for the first time from the test page to have a significantly lower predicted relevance rank. Note that because of the fact that we are not able to show each user its own personalized top k recommendations due to limitations of the online grocery app, it makes no sense to talk about our offline measures HR@k and NDCG@k in this context. The 40 items we showed on the test page are amongst these 4,134 items. Therefore, we tested potentially 40 different relevance scores per user, but not all users had a look at the page and certainly not all of the ones who did saw all 40 items. In the histograms in Figure 2 we only show the user-item combinations that were at least viewed[2]. These user-item combinations are split into two groups: the ones that were actually bought in the test set and the ones that were viewed but not bought. In both cases, we count the number of times an item from the test page was placed at position x of the recommended list of a user. As the number of purchased items makes up for only around 2% of the total views, both histograms show the number of elements in a bin as the percentage of the total number of elements. Absolute numbers are shown on top of each bar. It can be seen that of the user-item combinations with a purchase a much higher share has a predicted relevance rank below 100 than the combinations that were only viewed. Performing a two-sample Kolmogorov-Smirnov to test for the difference in distribution between these two groups, the null hypothesis of equal distribution is rejected with a p-value of 0.0008. We chose this non-parametric approach, as the distribution of the predicted ranks is hard to figure out due to the personalized rankings for each user. The median relevance rank for items that were actually bought is 256, in contrast to 327 for the items that were only viewed. This indicates the practical value of our model, showing that even in the context where all users get the same recommended items, we are able to tell which user-item combinations are most likely to occur.

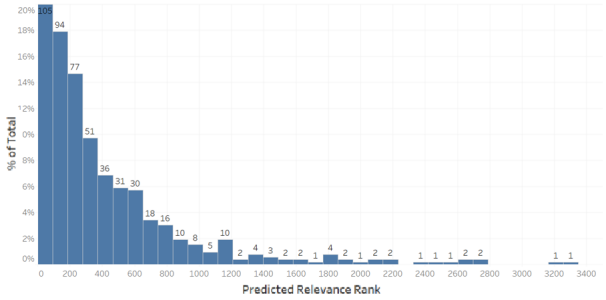
6 CONCLUSION

In this work we set out to estimate the personalized relevance of items that a user has never interacted with before. In our approach, we focused on the context of online grocery shopping where estimation can only be based on implicit feedback.

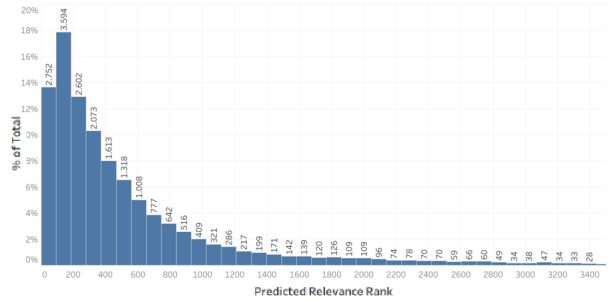
We combined the existing NeuMF and WMF frameworks into our NeuWMF model, which applies weighted implicit feedback to the domain of Neural Collaborative Filtering. The concept of weighting positive feedback stems from the idea that interacting with an item more regularly leads to more confidence that a user has an actual preference for it. We defined a weighting scheme based on the relative interaction frequency instead of absolute terms to prevent a bias towards fast-moving products. We showed that the NeuMF can be improved by including positive feedback weight instead of using a binary approach.

For items that a user never interacted with, we are more confident that this is a conscious choice for items that are more popular, leading to weights on negative feedback. We used negative sampling based on item popularity to incorporate weighted negative feedback in the model. This gives an extra boost to performance in terms of NDCG and HR when extra regularization is applied to prevent

[2]We regard items as viewed if they were positioned higher than the point where a user stopped scrolling down the page.



(a) Purchased user-item combinations during the test.



(b) Viewed, but not purchased during the test.

Figure 2: Histograms of predicted relevance score ranks, both for user-item combinations that were first bought during the test and those that were viewed but not bought. The histogram is cut off at 3500 for visualization and the number of user-item combinations in a bin is shown as a percentage of the total on the y-axis. Relevance is generally skewed to the left due to the fact that the average relevance of an item was included in the usefulness score that we used for picking items to show on the test page.

overfitting. Our NeuWMF model can also be estimated by L2-loss (which is used in WMF models) instead of binary cross-entropy loss (which is used in the NeuMF model) without harming performance too much.

We compared our NeuWMF to the combination of positive and negative feedback weights into a single Weighted Matrix Factorization (WMF) model. We implemented an element-wise ALS optimization method to show that the simultaneous weighting of both positive and negative feedback leads to improved performance of the RS. This is true both for evaluation in terms of HR and NDCG as well as for our modified NDCG evaluation rewarding only new items. The main source of improvement between the NeuMF and NeuWMF model is the implementation of positive weights.

In future work, it would be interesting to experiment further with different functions for the positive weights w_{ui} and the negative weights c_i. As an example, based on successful methods focusing on the order of purchases, like sequential event prediction [13], we would like to give more confidence to positive feedback that happened recently. Furthermore, the effect of adding biases on users and items to a Matrix Factorization model like in [9] to the WMF and NeuWMF model could be studied, as our user-item relevance scores are not directly comparable between users, since one user might generally score higher over items than another user.

REFERENCES

[1] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. 1999. *Modern Information Retrieval*. ACM.

[2] Linas Baltrunas and Francesco Ricci. 2009. Context-based splitting of item ratings in collaborative filtering. In *Proceedings of the 3rd ACM Conference on Recommender Systems (RecSys 2009)*. ACM, 245–248.

[3] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. 2019. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems (RecSys 2019)*. ACM, 101–109.

[4] Gabriel de Souza Pereira Moreira. 2018. CHAMELEON: a deep learning meta-architecture for news recommender systems. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys 2018)*. ACM, 578–583.

[5] Josef Feigl and Martin Bogdan. 2018. Neural Networks for Implicit Feedback Datasets. In *26th European Symposium on Artificial Neural Networks (ESANN 2018)*.

[6] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web (WWW 2017)*. ACM, 173–182.

[7] Xiangnan He, Han Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast Matrix Factorization for Online Recommendation with Implicit Feedback. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2016)*. ACM, 549–558.

[8] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *Proceedings of the 8th International Conference on Data Mining (ICDM 2008)*. IEEE Computer Society, 263–272.

[9] Christopher C Johnson. 2014. Logistic matrix factorization for implicit feedback data. In *Proceedings of the 28th International Conference on Advances in Neural Information Processing Systems (NIPS 2014), Workshops Track*. https://web.stanford.edu/~rezab/nips2014workshop/submits/logmat.pdf

[10] Alexandros Karatzoglou, Balázs Hidasi, Domonkos Tikk, Oren Sar-Shalom, Haggai Roitman, Bracha Shapira, and Lior Rokach. 2016. Workshop on Deep Learning for Recommender Systems. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys 2016)*. ACM, 415–416.

[11] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[12] Y. Koren, R. Bell, and C. Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37.

[13] Benjamin Letham, Cynthia Rudin, and David Madigan. 2013. Sequential event prediction. *Machine learning* 93 (2013), 357–380.

[14] Dawen Liang, Laurent Charlin, James McInerney, and David M Blei. 2016. Modeling user exposure in recommendation. In *Proceedings of the 25th International Conference on World Wide Web (WWW 2016)*. ACM, 951–961.

[15] Benjamin M. Marlin and Richard S. Zemel. 2009. Collaborative Prediction and Ranking with Non-Random Missing Data. In *Proceedings of the 3rd ACM Conference on Recommender Systems (RecSys 2009)*. ACM, 5–12.

[16] Aäron van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In *Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS 2013)*. Curran Associates, 2643–2651.

[17] Michael J. Pazzani. 1999. A Framework for Collaborative, Content-Based and Demographic Filtering. *Artificial Intelligence Review* 13 (1999), 393–408.

[18] Steffen Rendle. 2010. Factorization machines. In *Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2010)*. IEEE, 995–1000.

[19] Steffen Rendle and Christoph Freudenthaler. 2014. Improving pairwise learning for item recommendation from implicit feedback. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining (WSDM 2014)*. 273–282.

[20] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).

[21] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2015. *Recommender Systems Handbook* (2nd ed.). Springer.

[22] Guy Shani, David Heckerman, and Ronen I Brafman. 2005. An MDP-based recommender system. *Journal of Machine Learning Research* 6 (2005), 1265–1295.

[23] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM 2019)*. ACM, 1441–1450.

[24] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2015. Learning Hierarchical Representation Model for NextBasket Recommendation. In *Proceedings of the 38th International ACM Conference on Research and Development in Information Retrieval (SIGIR 2015)*. ACM, 403–412.