Web Intelligence 0 (2020) 1
IOS Press

# Active Learning Strategies for Solving the Cold User Problem in Model-Based Recommender Systems

Tomas Geurts [a], Stelios Giannikis [a], Flavius Frasincar [a,*]
[a] *Erasmus University Rotterdam, Burgemeester Oudlaan 50, 3062 PA, Rotterdam, The Netherlands*
*E-mail: tomasgeurts92@gmail.com, steliosgiannik@gmail.com, frasincar@ese.eur.nl*

**Abstract.** Customers of a webshop are often presented large assortments, which can lead to customers struggling finding their desired product(s), an issue known as choice overload. In order to overcome this issue, recommender systems are used in webshops to provide personalized product recommendations to customers. Though, model-based recommender systems are not able to provide recommendations to new customers (i.e., cold users). To facilitate recommendations to cold users we investigate multiple active learning strategies, and subsequently evaluate which active learning strategy is able to optimally elicit the preferences from the cold users in a matrix factorization context. Our model is empirically validated using a dataset from the webshop of de Bijenkorf, a Dutch department store. We find that the overall best-performing active learning strategy is PopError, an active learning strategy that measures the variance score for each item.

Keywords: Collaborative filtering, matrix factorization, active learning

## 1. Introduction

Modern-day consumers have made the transition from shopping at brick-and-mortar retailers with a limited assortment on display to shopping at webshops with practically unlimited assortment available. The most eminent example of such a webshop is the American on-line retailer Amazon.com, which offers over 480 million products. Even though Amazon.com has more products in its assortment compared to any other webshop, other webshops also have considerably larger assortments when compared to their offline counterparts.

Being able to present a wide and diverse assortment is important for each webshop. When webshops offer a large, varied assortment they can presumably better fit the individual needs of customers. However, when the size of the assortment grows substantially, a new issue arises for webshops. This issue is known as choice overload [1]. Choice overload implies that when customers are exposed to a considerably large assortment, they struggle with finding the desired products, and subsequently experience stress, anxiety, or other negative emotions. This could lead to dissatisfied customers, or even customers leaving the website and purchasing the desired product elsewhere. Hence, it is key for webshops to display products which are relevant for customers.

The challenge of offering a personalized, and thus relevant, assortment to customers gave birth to the concept of recommender systems (RS) [2]. RS's are effective when it comes to information filtering, news article recommendation, and numerous e-commerce related applications [3]. They provide personalized recommendations, which possibly increase the probability of a customer purchasing a product [4].

Interest in the field of RS's was boosted when Netflix initiated a contest where they challenged researchers and enthusiasts to outperform their RS at the time, Cinematch, by 10%. The performances of

[*]Corresponding author. E-mail: frasincar@ese.eur.nl.

the competing RS's were measured by the root mean square error (RMSE). Netflix promised one million dollars to the first team achieving this goal. On the 21st of September 2009 'BellKorr's Pragmatic Chaos' was the first team to outperform Cinematch by 10%.

Despite its importance, there are only a few works [5,6,7,8,9] that propose a solution regarding how to provide personalized recommendations to new users, also known as cold users. The reason why there are only few works that have come up with a solution to the cold user problem is evident: it is difficult to provide personalized recommendations to a user for whom there is no information available. However, being able to provide personalized recommendations to cold users is of vital importance for a webshop. Failing to deliver personalized recommendations can lead to cold users leaving the webshop, which in turn leads to the webshop missing out on a customer. In this scenario, a webshop not only loses one sale, but the customer might never return, hence missing out on many potential sales. Though, many works on RS's do not focus on this type of users. In fact, the majority of the works on RS's remove all users that have less than a certain number of ratings from their datasets. Researchers do this in order to ease the testing procedure of their RS's, because when more ratings are known from a user, the better the remaining ratings can be predicted. This is particularly the case for RS's utilizing collaborative filtering, which rely on comparing users by their past behaviour in order to provide recommendations.

Only a small number of existing works propose a solution to the cold user problem, usually by incorporating additional sources of information. Examples of additional sources of information regarding the cold users utilized to facilitate personalized recommendations to these cold users are social information [5], demographical information [6] (demographical information has been also used for improving collaborative filtering in general [10]), or queried item preferences [7,8,9]. For webshops, the first two options are not desirable as this information is not always available when a customer visits the webshop for the first time (demographical information), or customers are mostlikely not willing to share this information (social information). Yet, the latter option (querying item preferences) seems like a viable solution for webshops to facilitate providing personalized recommendations to cold users. We further explore such a solution in this paper, however, by taking a slightly different approach compared to the existing literature.

We recognize one serious problem for many RS's, being providing personalized recommendations to cold users. This in particular holds for RS's relying on matrix factorization, because a user who has not yet provided any information to the system cannot be compared to other users, and hence no personalized recommendations can be provided. Though, matrix factorization, has become increasingly popular because it is able to provide accurate recommendations [11]. Hence, it would be of particular interest to find a solution to the cold user problem for RS's which utilize matrix factorization. One of the most renowned RS's using matrix factorization is the model proposed by Koren et al. [12].

In this paper we focus on adapting an existing RS using matrix factorization, where we use the model from [12] as our reference, such that the cold user problem is addressed. This leads to the following research question: how can RS's using matrix factorization be adapted such that it can provide personalized recommendations to cold users?

In order to solve the problem posed in the research question, the model proposed by Koren et al. [12] is adapted. Our adaptation first elicits the preferences from the cold users with respect to a number of items, and subsequently incorporates the preferences of the cold users in the optimization procedure. What items are shown is decided by an item ranking, which depends on a particular active learning strategy. We produce multiple item rankings which rely on different strategies, and accordingly evaluate which active learning strategy elicits the preferences of the cold users best, ultimately leading to the most accurate recommendations. This work is an extension of our previous research presented in [13], where we propose four additional ranking methods (Misclassification Error, PopError, Variance, and PopVariance) that we evaluate against each other and our previously proposed methods.

## 2. Related Work

At the beginning of the 1990's the first works on RS's were published. During the same decade, research in the field of RS's slowly advanced into a full-grown field of research. [14,15,16]. Since the turn of the century, RS's have further been developed. For a complete overview we refer the reader to a number of surveys on recommender systems [17,2,18].

When RS's are not able to provide personalized recommendation to cold users, we refer to this as the cold user problem [19]. Collaborative filtering methods that do not incorporate additional information on the cold users, are not able to provide personalized recommendations to cold users, since there is no historical data available on these users. Hence, the RS can not compare them to other users because they have not rated or interacted with any items yet. This logic also applies to newly introduced items. The literature on RS's distinguishes between three types of RS's: content-based methods, collaborative filtering methods, and hybrid methods [2]. *Content-based* methods profile the users and items using available characteristics, e.g., user demographics or item specifications. Items are recommended to the users by comparing the characteristics of the previously obtained items with all other items, and recommending items that are relatively similar to the previously obtained items based on their characteristics. In [20] for example the items are songs, and for each item a large number of features are collected by a trained music analyst. These features capture not only the musical identity of a song, but also the many significant qualities that are relevant to understanding the musical preferences of listeners [20]. Thus, each song obtains its own genome. When a user listens to a number of songs, the RS recommends other, comparable songs to the user, based on the genomes of the songs the user previously listened to.

The second type of RS's is known as *collaborative filtering* methods. Collaborative filtering methods compare a user with other users and compute similarities between the users according to their past behaviour. This logic also applies to items. An overview of collaborative filtering methods is presented in [21]. An advantage of collaborative filtering methods is that they are not bound to a domain, and they are able to profile more complex relationships between users and items which are not identified by content-based methods [12]. This is the foremost reason why we focus on collaborative filtering methods in this paper.

The final type of RS's are *hybrid* methods, sometimes referred to as content-boosted collaborative filtering methods. Hybrid RS's utilize methods of both the content-based methods and the collaborative filtering methods, and combine these into one RS [22,23], levering the conveniences of both content-based RS's and collaborative filtering RS's. In the remainder of this paper we focus on RS's which rely on collaborative filtering.

### 2.1. Collaborative Filtering

The term collaborative filtering was introduced by Goldberg et al. [24]. They implemented collaborative filtering in their e-mail filtering system which was named Tapestry. This system was designed to filter e-mails received from mailing lists and newsgroup postings.

Collaborative filtering can be divided into two subcategories. On the one hand there are *memory-based* methods, also referred to as neighbourhood methods. Memory-based methods check the (full) user-item interaction matrix, the matrix with all user-item interactions, for users that are similar to the initial user, which is called the user neighbourhood. The first RS's were pure memory-based methods and scanned the full user-item interaction matrix for all pairwise similarities [25]. However, more recent memory-based methods were proposed that do not scan the full user-item interaction matrix, but use more advanced, and efficient methods to find users that are in the neighbourhood of the initial user [2]. Memory-based methods can even further be split into user-based methods [26], which compute similarities across users, and item-based methods [3], which compute similarities across items. Item-based methods are preferred over user-based methods, as in general there are more users than items, and hence item-based methods scale better.

In addition to memory-based methods, there are *model-based* methods, which, in contrast to memory-based methods, only utilize the user-item interaction matrix once to learn a predictive model, and subsequently use this model to predict the user-item interactions. Well-known model-based methods are Bayesian networks [27], cluster-based models [8], and matrix factorization [12]. From these three model-based methods, the latter method has become increasingly popular since the Netflix Prize competition.

Memory-based methods are generally easier to implement, and when the neighbourhood is not too large, they are also relatively scalable. On the other hand, the majority of the model-based methods provide better recommendations compared to their memory-based counterparts, and hence, dependent on the precise application, are often preferred over memory-based methods [11]. A relatively new, well-performing model-based method is matrix factorization, on which we elaborate in the next section.

## 2.2. Matrix Factorization

Matrix factorization, especially since the closing of the Netflix Prize competition, has become an increasingly popular model-based method used for collaborative filtering in RS's. Briefly, matrix factorization attempts to find lower dimensional vectors for both the users and items in which the common latent factors are captured. These lower dimensional vectors with latent factors are inferred from the user-item interaction matrix. The method is built on the belief that for both users and items there is a small amount of latent factors that fully capture the user's preference and the item's characteristics. Matrix factorization can be used to include both explicit feedback and implicit feedback [11]. There are multiple types of matrix factorization methods available. Among others, there are Regularized Singular Value Decomposition [28], Probabilistic Matrix Factorization [29], and Non-negative Matrix Factorization [30].

A recurring problem concerning matrix factorization methods is the fact that these methods only utilize the user-item interaction matrix, and can not incorporate additional information concerning the users or items when learning the latent factors of the users and the latent factors of the items. This implies that, when the user has not given (enough) feedback to the RS, providing personalized recommendations is very difficult.

## 2.3. Cold User Problem

When RS's are not able to provide personalized recommendation to cold users, we refer to this as the cold user problem [19]. Collaborative filtering methods that do not incorporate additional information on the cold users, are not able to provide personalized recommendations to cold users, since there is no historical data available on these users. Hence, the RS can not compare them to other users because they have not rated or interacted with any items yet. This logic also applies to newly introduced items.

With regard to the *cold user* problem, a possible solution is incorporating additional sources of information to enhance the user profile. Jamali and Ester [31] and Golbeck et al. [5] propose incorporating social information of cold users. Even though this seems like an effortless approach to gather more information on new customers, research has shown that (new) customers are reluctant to sharing their social information with webshops [32]. Another possible approach to include additional sources of information concerning cold users is including demographical information [6]. For this solution also holds that it is not desirable to ask for such information when a customer visits the webshop for the first time, since the webshop would like to refrain from making privacy-sensitive requests when the customer is new to, and not familiar with the webshop. The last approach to obtain additional information that we discuss here entails showing a number of items to the cold users and eliciting the preferences from their responses. In the literature this is regularly referred to as active learning methods [33]. This additional source of information is incorporated by, among others, Zhou et al. [9] and Yu et al. [8]. Even though it is not desirable to require a user to go through an interview process, this could be the least-harmful way of eliciting necessary preferences of users towards items in a short period of time, in order to provide personalized recommendations afterwards. Asking a cold user to provide demographical information (e.g., age, gender, place of residency) is generally considered as more privacy-intrusive compared to asking a cold user to provide her preferences towards a number of items.

Until a couple of years back, works concerning the cold start problem (cold user and cold item problem) focused in particular on the cold user problem. However, the *cold item* problem is possibly as precarious for webshops. In short, the problem entails that products which are recently added to the assortment are not recommended as often as products that have been around for a longer period of time since the current customer base has not been able to (frequently) buy these products yet. Hence, pure collaborative filtering methods less often recommend new products compared to products which are around for a longer period of time. However, contrary to the cold user problem, there are a number of solutions available for webshops to accommodate for the cold item problem. Existing content-boosted collaborative filtering methods [34] provide solutions to this issue when item attribute information is available, which can be assumed that is the case for webshops.

Even though the cold item problem is equally precarious as the cold user problem, for our research we focus on adapting RS's such that they are able to deal with the cold user problem. In the next section we further look into a number of adaptations which are able to deal with the cold user problem.

### *2.4. Ranking Problem*

There are several solutions for providing personalized recommendations to cold users. In particular, showing items to cold users in order to elicit their preferences, is a promising solution to this problem with regard to webshop applications. This solution is sometimes referred to as active learning for collaborative filtering [33]. Though, when requiring the opinion on a number of items from the cold users in a webshop, it is desirable to use as little time of the cold users as possible. In other words, we want to minimize the amount of time needed to learn the preferences from the cold users. Hence, it is crucial to show items to the cold users which have the most explanatory power, and give the most insight into the preferences of the cold users. In this section we review previous research regarding the ranking of items such that the most informative items are ranked highest.

Yu et al. [8] present a compelling argument why showing the "right" items to the cold users is of vital importance. Supposedly, cold users are not willing to spend much time on expressing their opinion on a number of items. This argument makes sense as customers that are new to a webshop should probably not be bothered with time-consuming questionnaires before being able to make use of the services that are offered within a webshop (e.g., a RS). Asking a new customer to fill in a questionnaire might increase the probability of a customer leaving the webshop before making a purchase. As a result of this, when eliciting the preferences of the cold users on the items, it is essential to show items which have large explanatory power. For example, when an item is shown to the cold users which has previously been rated by only one user, ratings from the cold users on this item do not give much information concerning their overall preferences, as it can only be compared with one other user.

Strategies that determine which items should be ranked highest are called active learning (AL) strategies [33]. We can split AL strategies into two groups, personalized and non-personalized AL strategies. The first type implies that personal information from the cold users are utilized to form an item ranking, and the second type does not use such information. We focus on the non-personalized AL strategies, as webshops rarely have personal information from cold users at their disposal. In [35] an overview of different types of AL strategies is provided. Yu et al. [8] propose an entropy-based strategy to determine the ranking of the items. Furthermore, Yu et al. [8] set the number of shown items to ten. In our research we experiment with this parameter, which is elaborated on in the upcoming section. Rashid et al. [7] also propose several AL strategies. Their best performing AL strategy combines a score inferred from the popularity of an item and an entropy-based score (we refer to this combined AL strategy as the PopEnt strategy in the remainder of this paper). For our AL strategies we follow a similar set-up as Rashid et al. [7]. However, our methodology differs from the methodology used by Rashid et al. on a number of points. First of all, next to the AL strategies evaluated by Rashid et al., we also evaluate the PopGini, PopError and the PopVar strategy, all variations on the PopEnt strategy, and the Gini, Misclassification Error and Variance strategies. Secondly, the model that provides the recommendations after the preferences of the cold users are elicited is different compared to Rashid et al., as we use a RS using matrix factorization, instead of nearest neighbors, to provide the personalized recommendations to the cold users. Finally, our dataset contains implicit feedback, while the dataset used by Rashid et al. contains explicit feedback.

### *2.5. Contribution*

Our contribution to the existing literature is two-fold. First of all, we adapt the model proposed by Koren et al. [12] to facilitate providing personalized recommendations to cold users, since personalized recommendations can possibly increase sales and customer loyalty. This is achieved through showing a number of items to the cold users and requiring their opinion regarding the shown items, in order to elicit the preferences from these cold users. The information gathered on the preferences of the cold users is included in the optimization of the objective function of the RS. We evaluate which AL strategy is able to gather the most information from the cold users. We investigate multiple AL strategies, from which a number are proposed earlier by Rashid et al. [7]. However, we also include five novel AL strategies, Gini, PopGini, Misclassification Error, PopError, and PopVar strategies. As this work is an extension of our previous research [13], two of these strategies, Gini and PopGini have been previously presented.

## 3. Methodology

In this section we elaborate on the model which we use for our research. This model is an adaptation to the model proposed by Koren et al. [12]. Our model specifically addresses the cold user problem for RS's using matrix factorization.

### 3.1. Notation

In a RS there are typically $N$ users (customers). The complete set of users is given by $U = \{u_1, u_2, ..., u_N\}$. Analogous to the notation of the users, there are $M$ items (products), where the complete set of items is given by $I = \{i_1, i_2, ..., i_M\}$. If a user $u$ interacts with an item $i$, this is denoted by $a_{ui}$. Interactions can be ratings on a scale from 1 to 5, in the case of explicit feedback (e.g., movie ratings), or binary ratings, in the case of implicit feedback (e.g., purchased an item, looked at an item, etc.). The complete set of interactions between all users and all items, the user-item interaction matrix, is given by $A \in \mathbb{R}^{N\times M}$. The domain of $a_{ui}$ is given by $a_{ui} \in \{0, 1\}$. Since the data we use for the RS is implicit feedback, we have chosen to limit the values of the interactions to either 0 or 1. In our dataset, the value 1 indicates one or multiple purchases, and 0 indicates that the item is returned. We follow the reasoning of Schafer et al. [4]: if a product is purchased twice and only one of the two products is returned, we record this as a 1, as the sum of the interactions is positive. It holds that the majority of the users only interacted with a subset of the complete set of items $I_u \subseteq I$, where $I_u$ is defined as the subset of items interacted with by user $u$. Hence, in general we expect to observe only a small number of 1's and 0's per user.

### 3.2. Matrix Factorization

Matrix factorization attempts to map both the user matrix and the item matrix to the same latent space. The user latent factor matrix, $P \in \mathbb{R}^{F\times N}$, can be regarded as the preferences of users towards the different latent factors, while the item latent factor matrix, $Q \in \mathbb{R}^{F\times M}$, can be regarded as the resemblance of items with the different latent factors. Furthermore, it holds that $F \ll \min(N, M)$. By multiplying the matrix $P$ and the matrix $Q$, an approximation to the user-item interaction matrix $A$ is computed. It should be noted that $A$ can contain missing values, i.e., since no user has interacted with all the items (most users have only interacted with a very small amount of items).

Explaining the observed interactions only by the inner product of $P$ and $Q$ is not sufficient according to Koren et al. [12], since the interactions can also partially be explained by effects corresponding to a particular user or particular item. Therefore, we include three biases: a global bias (represented by $\mu$), a user bias (represented by $b_u$), and an item bias (represented by $b_i$). For the values of the latent factors corresponding to user $u$, $p_u$, and to item $i$, $q_i$, the approximation to the user-item interaction $a_{ui}$, is given by,

$$a_{ui} \approx \hat{a}_{ui} = \mu + b_u + b_i + p_u^T q_i. \tag{1}$$

where $\hat{a}_{ui}$ is the approximation to the initial user-item interaction $a_{ui}$.

Once $P$ and $Q$ are known, finding $\widehat{A}$ is straightforward. However, the challenge lies in finding the latent factor matrices $P$ and $Q$. Typically, this is achieved using Singular Value Decomposition (SVD). SVD was used to perform latent semantic analysis in the early 1990's [36]. Subsequently, Paterek [28] and Koren [11] applied a technique similar to SVD successfully to matrix factorization.

However, as ordinary SVD is not feasible when we are dealing with a sparse user-item interaction matrix, this technique is adapted in order to deal with missing values. A possible solution to this problem is fitting the model only to the observed interactions. Even though this solves the previous problem, it causes the model to be very susceptible to overfitting. Kim and Yum [37] use imputation to overcome possible overfitting, but this in turn leads to more expensive computations. More recently, several works [11,28,38] fit the model only on the observed interactions, but they include a regularization parameter to avoid the model from overfitting. We adopt the same approach, but, in contrast to Koren et al. [12], who only include one global regularization parameter, we choose to include two separate regularization parameters, one for the bias terms and one for the latent factors, which should give more flexibility to the model. This leads to the following model,

$$\begin{aligned} \min_{p,q,b} \sum_{(u,i)\in K} & (a_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \\ & \lambda_1 (b_u^2 + b_i^2) + \lambda_2 (||p_u||^2 + ||q_i||^2), \end{aligned} \tag{2}$$

where $K$ is the set containing the interactions that are included in the training set, $\lambda_1$ is the regularization parameter for the bias terms, and $\lambda_2$ is the regularization parameter for the latent factors. To convert the predictions made by the model to a binary variable, we use the following logistic function,

$$\hat{a}^*_{ui} = \left[ \frac{1}{1 + \exp(-\hat{a}_{ui})} \right], \tag{3}$$

in this expression $\hat{a}_{ui}$ is the predicted interaction for user $u$ with item $i$ and $\hat{a}^*_{ui}$ is the dichotomized predicted interaction.

There are different algorithms which can be used to solve Equation 2. Commonly applied is Stochastic Gradient Descent (SGD), which considers one observation at a time, lowering the computations per iteration. In short, this algorithm implies that for each observed rating included in the training set the prediction error is computed. Formally,

$$\epsilon^{\text{pred}}_{ui} = a_{ui} - \hat{a}^*_{ui}. \tag{4}$$

Accordingly, $b_u$, $b_i$, $p_u$, and $q_i$ are updated using the following rules,

$$b_u \leftarrow b_u + \gamma_1(\epsilon^{\text{pred}}_{ui} - \lambda_1 \cdot b_u), \tag{5}$$

$$b_i \leftarrow b_i + \gamma_1(\epsilon^{\text{pred}}_{ui} - \lambda_1 \cdot b_i), \tag{6}$$

$$p_u \leftarrow p_u + \gamma_2(\epsilon^{\text{pred}}_{ui} \cdot q_i - \lambda_2 \cdot p_u), \tag{7}$$

$$q_i \leftarrow q_i + \gamma_2(\epsilon^{\text{pred}}_{ui} \cdot p_u - \lambda_2 \cdot q_i), \tag{8}$$

where $\gamma_1$ and $\gamma_2$ are the learning parameters.

### 3.3. *Strategies for Item Ranking*

For a cold user $k$, the user vector $u_k$ is an empty vector because this user has not interacted with any items so far. Since we have no information available concerning this user, we cannot provide any personalized recommendations to this user. To be able to provide such recommendations, we should elicit her preferences towards a number of items. In other words, we would like to know the true values for $\{a_{k1}, ..., a_{kj}\}$, where $j$ is a finite number for which holds that $j \ll M$.

The goal is to elicit the preferences of the user towards the $j$ products, which lead to the highest recommendation quality. In our research we try different values for $j$ as explained in the evaluation. We present ten strategies which are used to generate the item rankings from which the top-ranked items are shown to the cold users. All the AL strategies that we propose do not account for user-specific information and require each user to give their opinion on the same set of items. As mentioned before, these AL strategies are considered as non-personalized AL strategies [35]. A number of AL strategies that we propose (the random strategy, the entropy strategy, the popularity strategy, and the PopEntropy strategy) have been investigated in other works [39,7] in a different context. Five novel AL strategies are proposed in this paper: the Gini strategy, based on the Gini impurity measure, the PopGini strategy, which is a combination of item popularity and Gini impurity score, the Misclassification Error strategy, the PopError strategy, which is a combination of the Popularity strategy and the Misclassification Error, and finally the PopVar strategy, which is a combination of the Popularity strategy with the Variance strategy introduced by [40], but has not yet been applied for implicit feedback systems.

For the Random strategy, Gini strategy, Entropy strategy, Misclassification Error strategy, Variance strategy PopGini strategy, PopEnt strategy, PopError, and PopVar strategy only items for which the number of interactions is larger than ten are considered. If we do not impose this threshold, items that have only been interacted with a small number of times ($< 10$) are possibly ranked high. We set this value to ten because for this value we find the best trade-off between the number of items eligible to be selected using the random strategy and the number of available interactions per item.

#### 3.3.1. *Random Strategy*

We first propose the random strategy, which entails a random selection of items to form the item ranking. This AL strategy is incorporated as a baseline for the other AL strategies, as we expect the other AL strategies to outperform a randomly assembled item ranking.

The advantage of the random strategy is the fact that the ranking of the products is at random and hence, the shown products are by definition uncorrelated. Furthermore, all products from the assortment have a probability of being presented, in contrast to the other AL strategies which are presented in this section.

#### 3.3.2. *Popularity Strategy*

The second AL strategy that we propose is the popularity strategy, which can be seen as a single-heuristic attention-based AL strategy. With a single-heuristic attention-based AL strategy we imply an AL strategy that only depends on a single heuristic whose score is based on the amount of 'attention' an item receives (i.e., the popularity). This AL strategy implies that the most popular items are ranked highest in the item ranking. With popular items we denote items that have been interacted with by the largest number of distinct users.

An advantage of this AL strategy is that the items which are ranked high are items which many users have interacted with. On the other hand, there are two severe disadvantages to this AL strategy.

The first disadvantage is the fact that many items which are ranked high can be relatively similar. For example, it could be possible that the three most popular (and thus highest ranked) items are all t-shirts from the same brand. Hence, once we know the opinion of the customer on the first of these three items, the following two responses do not provide much additional insight in the preferences of the cold user (but do require an effort from this cold user).

The second disadvantage entails that this AL strategy only elicits the preferences from the cold users on a number of popular items. This could lead to the problem that the RS also recommends more popular items as opposed to less popular items.

#### 3.3.3. *Gini Strategy*

Next, we propose the Gini strategy, which uses the Gini impurity measure to compute the item ranking. This AL strategy is considered as a single-heuristic uncertainty-based AL strategy. With an uncertainty-based AL strategy we imply an AL strategy that is based on an impurity-measure, computing the 'uncertainty' for each item. The Gini impurity for item $i$ is given by,

$$\text{Gini}(i) = 1 - \sum_{j \in 0,1} (p(j|i))^2, \tag{9}$$

where $p(j|i)$ is the relative frequency of a positive interaction ($j = 1$) or negative interaction ($j = 0$) with item $i$. Accordingly, the items are shown in descending order to the user. A high Gini impurity for item $i$ implies that the ratio between positive interactions and negative interactions is (relatively) balanced. Intuitively, items with high Gini impurity are good in splitting the set of users into equally sized groups.

The Gini strategy ensures that the items that are shown are able to optimally split the set of users, which is considered advantageous. Specifically, the Gini strategy highly ranks items which have received much contrasting feedback from the users. As a result of this contrasting feedback, the RS is rather unsure about the user's opinion on these items. Subsequently, when a cold user expresses her opinion on this item, this gives useful information regarding the cold user's preferences [35].

#### 3.3.4. *Entropy Strategy*

Another commonly applied AL strategy is based on the entropy measure [7,41]. Similar to the Gini strategy, the Entropy strategy is also a single-heuristic uncertainty-based AL strategy. We also include this AL strategy in our research, where we use Shannon's entropy as the designated entropy measure. The entropy for item $i$ is computed accordingly,

$$\text{Entropy}(i) = - \sum_{j \in 0,1} p(j|i) \log_2 p(j|i), \tag{10}$$

where $p(j|i)$ is the relative frequency of a positive interaction ($j = 1$) or negative interaction ($j = 0$) with item $i$. Accordingly, the items are shown in descending order to the user. A high entropy for item $i$ implies that the ratio between positive interactions and negative interactions is (relatively) balanced. Intuitively, items with a high entropy are good in splitting the set of users into equally sized groups.

#### 3.3.5. *Misclassification Error Strategy*

Another new strategy being proposed, is the Misclassification Error strategy which from now on will be called "Error". Error is a static non-personalized single-heuristic uncertainty-based strategy used for the item ranking. The Error for an item $i$ is given by,

$$Error(i) = 1 - \max_{j \in \{0,1\}} (p(j|i)) \tag{11}$$

where it is one minus the maximum relative frequency of a positive ($j = 1$) or a negative ($j = 0$) interaction.

#### 3.3.6. *Variance Strategy*

Next, the Variance strategy [40], a single-heuristic uncertainty-based AL strategy, aims at finding items with the largest variance in their interactions. Items with high score on variance, are considered uncertain, as the system does not have a clear rating for those items. The formula for the Variance strategy is given by,

$$Variance(i) = \frac{1}{|U_i|} \sum_{u \in U_i} (r_{ui} - r_i)^2 \tag{12}$$

where $|U_i|$ is the number of users $u$ who have interacted with the item $i$, $r_{ui}$ is the interaction (either 1 or 0) for the item $i$. Finally, $r_i$ is the mean interaction for the item $i$, taking into consideration all the users who have interacted with the item $i$, either positively or negatively.

### *3.3.7. PopGini Strategy*

This AL strategy entails a linear combination of the scores produced by the popularity strategy (see Section 3.3.2) and the Gini strategy (see Section 3.3.3). This AL strategy can best be described as a static combined-heuristic AL strategy. The popularity score (i.e., interaction frequency of an item) tends to dominate the PopGini score because of several outliers, items for which many distinct users have interacted with them. This leads to the popularity strategy and PopGini strategy being relatively similar. To overcome this issue (as we prefer that both scores are relatively balanced), we take the logarithm with base 10 of the popularity score. We choose for a log transformation since the distribution of the item frequencies are rightly-skewed, i.e. many items which have only a few interactions and a small amount of items that have many interactions.

This AL strategy both accounts for popular items, where many users have interacted with, and items which are relatively different (in terms of feedback given by the users), such that more information can be elicited from the cold users. The importance weights for the two components are optimized based on the training data.

### *3.3.8. PopEnt Strategy*

The next AL strategy is similar to the PopGini strategy presented in Section 3.3.7, and also implies a linear combination of two scores. For this AL strategy, we use a linear combination of the popularity strategy (see Section 3.3.2) and the entropy strategy (see Section 3.3.4). The PopEnt strategy can best be described as a static combined-heuristic AL strategy. Similar to the PopGini strategy, the popularity score tends to dominate the PopEnt score. This leads to the popularity strategy and PopEnt strategy being relatively similar. To overcome this issue we take the logarithm with base 10 of the popularity score.

This AL strategy, similar to the PopGini strategy, both accounts for popular items, where many users have interacted with, and items which are relatively different, such that more information can be elicited from the cold users. This AL strategy is, as mentioned before, relatively similar to the PopGini strategy. However, it is of particular interest whether the Gini impurity measure or entropy measure performs better when combined with the popularity score. The importance weights for the two components are determined in the evaluation.

### *3.3.9. PopError Strategy*

The PopError strategy is another novel strategy for addressing the cold-user problem, which is a combination of the Popularity and the Error strategies that tries to exploit the advantages of both the Popularity and the Error strategy. A linear combination of these two strategies is taken and weights are assigned to each of the strategies. To deal with the issue regarding the popularity strategy, the logarithm with base 10 is once again taken for the popularity strategy (see sections 3.3.7 and 3.3.8).

### *3.3.10. PopVar Strategy*

The PopVar strategy, a combination of the Popularity strategy and the Variance strategy, is considered a non-personalized combined-heuristic strategy (all the considered AL strategies in this work are non-personalized). Like the previous combined-heuristic strategies, PopVar takes into account the frequency of the items, therefore exploiting the most popular items that many users have purchased. Next to the Popularity strategy, the Variance strategy scores high items with high variance in their interactions, meaning items with very diverse interactions that are neither popular nor unpopular. The combination of these two strategies is achieved by assigning weights to each of the strategies, which will be explained in detail in the evaluation section. Like the other combined heuristic strategies, a linear combination is made and the logarithm with base 10 of the popularity score is taken to deal with the issue of the popularity score dominating the Variance score.

## 4. Evaluation

In this section we explain the evaluation of the model proposed in this paper. Before presenting and discussing our results we first introduce the dataset that is used for our research. Subsequently, we present the metric by which our model is evaluated, and the set-up that is used to perform this experiment. We then finalize with presenting the results and discussing these.

### *4.1. Dataset*

To empirically validate our model we use a dataset[1] provided by de Bijenkorf, a Dutch department store

[1]Dataset is available from `http://tinyurl.com/z8mqele`.

with a large webshop. The webshop has over 100,000 unique visitors each day, over a quarter million euro sales each day, and more than 200,000 unique products currently in their assortment.

The dataset contains all interactions between customers and products (i.e., purchases and returns) from 14 July 2015 till 13 July 2016. There are 563,495 unique customers and 242,020 unique products in our dataset, and in total the dataset contains 2,563,878 unique interactions between a customer and a product. We have chosen for dichotomized interactions, and hence the interactions are assigned a 1 when the number of times product $i$ is purchased by customer $u$ is larger than the number of times product $i$ returned by customer $u$ and a 0 when these two values are equal.

Out of the 2,563,878 interactions, 2,159,538 interactions are positive interactions (i.e., are assigned a 1) and 404,340 interactions are negative interactions (i.e., are assigned a 0). If a customer has not purchased (and hence also not returned a product), the interaction between the customer and the product does not exist (i.e., it is missing from the user-item interaction matrix). This implies that the user-item interaction matrix is very sparse. Each customer has interacted with 4.55 items on average, while each item has been interacted with by 10.59 customers on average. Finally, in this dataset all customer identifiers are anonymized.

### 4.2. Set-up and Model Parameters

To evaluate the performance of the model proposed in this paper we include the score on the most-widely used metric in the field of RS's. This metric indicates how the proposed models score with regard to the recommendation quality. The metric which we report is the root mean square error (RMSE), which is given by,

$$RMSE = \sqrt{\frac{\sum_{(u,i)\in T}(a_{ui} - \hat{a}^*_{ui})^2}{|T|}}. \tag{13}$$

where $T$ is the test set, $a_{ui}$ is the interaction of user $u$ with item $i$ and $\hat{a}^*_{ui}$ is the dichotomized predicted interaction of user $u$ with item $i$. For the RSME holds that the lower its value, the better the recommendation quality of the proposed model. We choose to only report the RMSE, and not the MAE, as the RMSE and the MAE are equivalent (up to a square root) when using binary values as the input for user-item interaction matrix.

The goal of our experiment is to evaluate which AL strategy elicits the most information from the cold users such that the RS is able to provide the best recommendations, in terms of the RMSE, to these cold users. We elicit the information from the cold user by showing them a number of items and requiring their opinion on these items. We evaluate the AL strategies for different numbers of items shown to the cold users: 10, 25, 50 and 100.

We use the following experimental set-up: 25% of the total set of users is randomly selected to be a cold user. Similarly to other works that attempt to elicit information from cold users, this number is picked somewhat arbitrarily [39]. All the interactions of the non-cold users are included in the training set. Additionally, all the interactions of the cold users with items that are included in the item ranking under consideration, under the condition that the cold user has interacted with the item shown to her, are also included in the training set (e.g., if cold user $u$ has only interacted with one of the items shown to her, only that particular interaction is included in the training set). The remaining interactions of the cold users are included in the test set. Only cold users that have interacted with at least one item in the ranking under consideration (i.e., items shown to the cold user) are included in the training and test set. If a cold user has not interacted with any of the items shown to her (this can be interpreted as the cold user having no opinion on any of the items from the item ranking shown to her), this particular cold user is excluded from both the training set and test set. We do this because if this user is still included, we would be providing recommendations to a cold user for which the ranked items did not elicit any information from this particular user, and hence the recommendations are not personalized. The intuition behind this set-up is that previous to making recommendations to the cold users, the only information we have at our disposal is the information on the interactions of the previous (non-cold) users and the preferences of the cold users with respect to the items that are shown to them (i.e., the items from the item ranking under consideration).

Before conducting our experiments we first tune the model parameters. During both the model parameters tuning and the experiments, we set the maximum number of iterations for the optimization procedure to 50. To find the optimal number of factors to include in the matrix factorization, and the optimal values for the regularization parameters, we use a grid search. In order to validate the results of the grid search, we perform 10-fold cross-validation on the complete set of interactions for each parameter combination. We find that the optimal number of factors is 200, the optimal value for $\lambda_1$ (regularization parameter for the bias terms) is $1 \times 10^{-7}$, and the optimal value for $\lambda_2$ (regularization parameter for the latent factors) is $1 \times 10^{-6}$. For the remainder of the parameter tuning and our later experiments we continue to use these values. To provide some more insight in the model parameters, we also include a sensitivity analysis. When adjusting $\lambda_1$ to its next-best value, being $1 \times 10^{-8}$, we observe that the RMSE decreases by $0.29\%$. Similar, for $\lambda_2$ the next-best value is $1 \times 10^{-5}$, but this implies a decrease of $2.26\%$ in terms of RMSE, which is almost 10 times as much as the decrease in RMSE for adjusting $\lambda_1$ to its next-best value. Adjusting the number of factors has the least influence on the performance in terms of RMSE, the next-best value for the number of factors to include in the matrix factorization is $150$ factors, and this causes a decrease of $0.02\%$ in terms of RMSE. Hence, we conclude that the RMSE depends most on $\lambda_2$, followed by $\lambda_1$, and then the number of factors included.

Next to the model parameters, we also compute the optimal weights for the two components of the PopGini, PopEnt, PopError and PopVar score. To obtain the optimal weights, the RMSE of each combination of weights using the initial experimental set-up is computed. For each combination we take the mean of the RMSE for 10 and 100 items shown to the cold users. The range of the weights are the values between $0.1$ and $1$, with step size $0.1$. Hence, in total there are $100$ unique combinations of weights. We use that $25\%$ of the users is selected to represent cold users. The optimal weights are $0.9$ for the popularity component and $1$ for the entropy component.

We have implemented our model and the AL strategies for computing the item rankings in Python, and ran our experiments on a C4 instance of Amazon EC2 with 8 vCPU's (High frequency Intel Xeon E5-2666 v3 Haswell processors) and 15GB of RAM. The RS which is included in our model is built using the machine learning framework GraphLab Create developed by Turi. The utilized machine learning framework is built in Python and backed by a C++ engine. This framework facilitates building and deploying machine learning applications at scale, which is desirable when working with very large datasets.

### 4.3. *Results*

In this subsection we evaluate the different AL strategies on their performance in terms of the RMSE. The results are presented in Figure 1 and Table 1. From Figure 1 we can conclude that when 25 items are shown to the cold users, the random strategy is the best-performing AL strategy in terms of the RMSE. It is surprising that the random strategy significantly outperforms the other AL strategies for 25 items shown, however, if we look at the remainder of the results, we observe that the random strategy varies substantially in its performance. However, it is closely followed by the PopGini strategy with a difference of 0.0372 between the two strategies. For 10 items, the Variance strategy outperforms the rest of the strategies, closely followed by the PopVar strategy. For 50 items, the PopError strategy dominates while for 100 items, the PopGini strategy performs best.

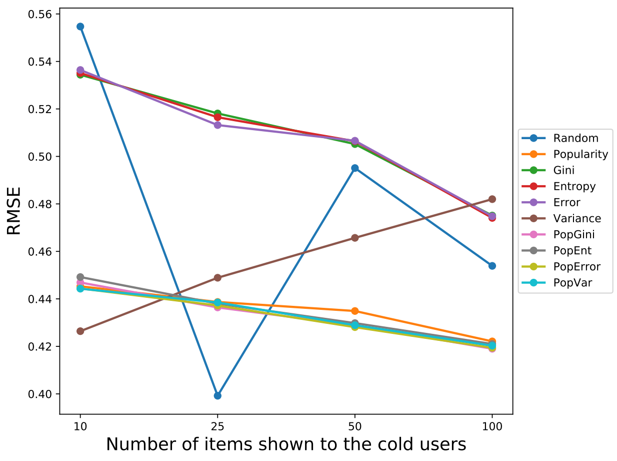

Fig. 1. Graph visualization of the RMSE for different numbers of shown items per AL strategy.

Surprisingly, the impurity measure-based AL strategies are the worst-performing AL strategies. The lowest RMSE is achieved by the random strategy for 25 items.

### 4.4. *Discussion*

Having presented the results in the previous section, in this section we further investigate the results and discuss their implications.

When we take a closer look at Figure 1 we can make a number of observations. First, for the random strategy the results vary a lot for different number of items shown to the cold users, which makes sense as the random strategy proposes random items to gauge user preferences, some of these items are useful while others are not. Also, differently than for the impurity-based measures the Variance strategy leads to an increasing function, which is probably due to the fact that the variance measures the spread of the values and depends on our class encoding, i.e., 1 for a liked item and 0 for a disliked item, while the impurity measures are based on the class distribution (we also noticed that the changes in variance are smaller than in the considered impurity measures as we increase the number of items shown to the cold users, making more difficult the selection of the most relevant items).

Second, the popularity strategy performs better than expected surpassing in all situations the Entropy, Gini, and Classification Error strategies. This is probably due to the fact that many users have a (strong) opinion on popular items and hence, the items are very informative regarding the users preferences.

Third, the impurity measure-based AL strategies are the worst-performing AL strategies. This seems surprising, especially considering that these AL strategies perform worse than the random strategy except when the number of shown items is 10. A possible explanation could be that the bad performance of these AL strategies is due to the fact that odd items are being ranked high. Items that have almost equal purchase and return rates (and thus rank high when using these strategies to generate the item rankings) are most of the time items where there is some deficiency or malfunctioning, and hence, these items might not be a good reflection of the true preference of users towards an item.

At last, we observe that, according to the number of shown items, different strategies perform best. In two situations, when the number of shown items is 50 and 100, the PopError and the PopGini strategies are the best-performing AL strategies since they achieve the lowest RMSE. Overall, the best-performing AL strategy is the PopError strategy, since this AL strategy achieves the lowerst mean RMSE over all number of items shown to the cold users. This observation is plausible as this AL strategy depends both on the popularity strategy, which performs well on average as in every situation outperforms the Gini and the Entropy strategy, and on the Error strategy. The combination of these two, appears to be the overall best-performing AL strategy.

Table 1

Tabular representation of the RMSE for different numbers of shown items per AL strategy (best values per column are given in bold font).

| ***Ranking*** | **Number of items shown** | | | | |
|---|---|---|---|---|---|
| ***Strategies*** | ***10*** | ***25*** | ***50*** | ***100*** | ***Mean*** |
| Random strategy | 0.5547 | **0.3992** | 0.4951 | 0.4539 | 0.4757 |
| Popularity strategy | 0.4453 | 0.4387 | 0.4349 | 0.4221 | 0.4353 |
| Gini strategy | 0.5344 | 0.5181 | 0.5052 | 0.4751 | 0.5082 |
| Entropy strategy | 0.5351 | 0.5165 | 0.5064 | 0.4741 | 0.5080 |
| Error strategy | 0.5364 | 0.5132 | 0.5066 | 0.4748 | 0.5078 |
| Variance strategy | **0.4264** | 0.4489 | 0.4657 | 0.4820 | 0.4558 |
| PopGini strategy | 0.4469 | 0.4364 | 0.4289 | **0.4190** | 0.4328 |
| PopEnt strategy | 0.4492 | 0.4380 | 0.4298 | 0.4210 | 0.4345 |
| PopError strategy | 0.4444 | 0.4372 | **0.4281** | 0.4194 | **0.4323** |
| PopVar strategy | 0.4443 | 0.4385 | 0.4289 | 0.4203 | 0.4330 |

## 5. Conclusions

Nowadays, RS's are widely used in webshops. They are able to provide personalized recommendations to customers of the webshop. One of the most popular methods used for RS's is matrix factorization. Though, RS's using this method are not able to provide personalized recommendations to cold users. In this paper we present an adaptation to the model proposed by Koren et al. [12], such that this type of RS is able to provide personalized recommendations to cold users. To be able to provide personalized recommendations to cold users, we opt to adapt the model proposed by Koren et al. [12], by showing a number of items to the cold users, to elicit the preferences from these cold users. Using this information on the cold users we are able to provide personalized recommendations. In our research we evaluate different strategies to produce the item rankings. From the item rankings a number of the highest ranked items are shown to the cold users. We evaluate which AL strategy places the most informative items on top of the item ranking, such that as much information as possible is elicited from the preferences of the cold users, which subsequently leads to the most accurate recommendations being provided.

The results show that the AL strategy which performs best is dependent on the number of items that are shown to the cold users. When we only show ten items to the cold users, we observe that the Variance strategy outperforms the other nine strategies in terms of the RMSE. When the number of shown items to the cold users increases to 25, then the Random strategy outperforms the other strategies. Next, for 50 shown items, the PopError strategy performs best, by combining the Popularity strategy with the Error strategy. Finally, when 100 items are shown to the cold users, the proposed PopGini strategy, a combination of the Popularity strategy and the novel Gini strategy outperforms all other AL strategies. Overall, we conclude that the PopError strategy is the best-performing AL strategy, as it achieves the lowest mean RMSE.

Concluding, in this paper we have taken a specific approach to facilitating personalized recommendations to cold users. We do this by showing the cold users a number of items and asking for their opinions on these items. Using this information, personalized recommendations are provided to the cold users. This paper contributes to the literature by proposing several new AL strategy, named the Gini, PopGini, Error, PopError and the PopVar strategy, using a different type of dataset compared to other works regarding this topic, and using a RS relying on matrix factorization to provide the personalized recommendations to the cold users. In the remainder of this section we discuss possible directions for future research.

For future work we distinguish between three directions. First of all, the data collected for our research originates from the webshop of de Bijenkorf. In our research, to determine the opinion of a customer with respect to an item, we take into account purchases and returns. We assumed that a positive interaction is only dependent on purchases and returns. Though, it can be argued that this interpretation of a positive and a negative interaction is subjective. Additionally, the user-item interaction matrix is very sparse, as there are many customers who have only interacted with a small number of products. For example, a little less than $46\%$ of all customers have only interacted with one product. In comparison, in the Netflix dataset $50\%$ of the users have rated at least 100 movies. From this we can conclude that on the majority of the customers in our dataset, there is only a minimum amount of information available. This is obviously not beneficial for the recommendation quality. In order to obtain more information on each customer, additional implicit feedback of users should be gathered. In our research we only use purchases and returns because of data availability at de Bijenkorf. However, if at the researchers' disposal, on-line behaviour of customers can also be taken into account. Incorporating this additional information leads to a more dense user-item interaction matrix. The advantage of having a dense user-item interaction matrix is the fact that it includes more information on the customers and hence, more accurate recommendations can be made. However, it comes at the cost of computational efficiency.

With regard to the methodology used in our research, there are a number of alterations possible. First, it would be of interest to include other AL strategies. Other AL strategies could be based on different impurity measures (e.g., entropy0 [42]), or possibly follow a completely different approach (e.g., greedy extend [43]). Moreover, in our research we use that the items shown to the cold users are the same for each cold user, regardless of the opinion of the cold user. In this setting, the opinion of a cold user on the first item shown to her is not used when determining which item to show next. It might be advantageous to make the item shown to the cold user dependent on her opinion on the previously shown item. This type of AL strategies are also known as personalized AL strategies. Secondly, in the current set-up the model is learnt during the training phase using the errors on the predictions of all users, both non-cold and cold users. This most-likely ensures the best overall performance in terms of the RMSE. However, if we prioritize the predictions made for the cold users over the non-cold users, there are some alterations possible with regard to the way that the matrix factorization method is applied. Instead of treating all errors equally, it could be investigated whether adding a cost to the errors of the cold users would improve the recommendations to the cold users.

Our current experimental set-up enables us to execute our research using only the initial dataset, without requiring additional feedback from the cold users. This particular set-up eases the execution of our research, as we do not actively need to gather additional feedback from the cold users. However, this set-up also has a few drawbacks. Some cold users are excluded from the test set because they have not interacted with any of the items shown to them. Especially when the number of items shown to the cold users is small (which in a webshop application would be the case), many cold users are excluded from the test set. This heavily affects the results as test sets differ substantially for different AL strategies and number of shown items. Furthermore, the information gathered per cold user can differ substantially, as it could be very well possible that the first cold user has interacted with the first half of the items shown to her, while the second cold user has interacted with the second half of the items shown to her. A possible solution would be performing a live user study in which the cold users are asked to provide their opinion on the highest ranked items from the different AL strategies. In this set-up we elicit the preferences from all cold users on all highest ranked items from the different AL strategies, leading to the same amount of information gathered for each cold user. However, this would require a substantial effort from cold users.

## References

[1] S.S. Iyengar and M.R. Lepper, When choice is demotivating: Can one desire too much of a good thing?, *Journal of Personality and Social Psychology* **79**(6) (2000), 995.

[2] G. Adomavicius and A. Tuzhilin, Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions, *IEEE Transactions on Knowledge and Data Engineering* **17**(6) (2005), 734–749.

[3] G. Linden, B. Smith and J. York, Amazon.com recommendations: Item-to-item collaborative filtering, *IEEE Internet Computing* **7**(1) (2003), 76–80.

[4] B.J. Schafer, J. Konstan and J. Riedl, *Recommender systems in e-commerce*, in: Proceedings of the 1st ACM Conference on Electronic Commerce (EC 1999), ACM, 1999, pp. 158–166.

[5] J. Golbeck, J. Hendler et al., *FilmTrust: Movie recommendations using trust in web-based social networks*, in: Proceedings of the 3rd IEEE Consumer Communications and Networking Conference (CCNC 2006), Vol. 96, IEEE, 2006, pp. 282–286.

[6] M.J. Pazzani, A framework for collaborative, content-based and demographic filtering, *Artificial Intelligence Review* **13**(5–6) (1999), 393–408.

[7] A.M. Rashid, I. Albert, D. Cosley, S.K. Lam, S.M. McNee, J.A. Konstan and J. Riedl, *Getting to know you: Learning new user preferences in recommender systems*, in: Proceedings of the 7th International Conference on Intelligent User Interfaces (IUI 2002), ACM, 2002, pp. 127–134.

[8] K. Yu, A. Schwaighofer, V. Tresp, X. Xu and H.-P. Kriegel, Probabilistic memory-based collaborative filtering, *IEEE Transactions on Knowledge and Data Engineering* **16**(1) (2004), 56–69.

[9] K. Zhou, S.-H. Yang and H. Zha, *Functional matrix factorizations for cold-start recommendation*, in: Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2011), ACM, 2011, pp. 315–324.

[10] M.G. Vozalis and K.G. Margaritis, On the enhancement of collaborative filtering by demographic data, *Web Intelligence and Agent Systems* **4**(2) (2006), 117–138.

[11] Y. Koren, *Factorization meets the neighborhood: A multifaceted collaborative filtering model*, in: Proceedings of the 14th ACM International Conference on Knowledge Discovery and Data Mining (KDDM 2008), ACM, 2008, pp. 426–434.

[12] Y. Koren, R. Bell and C. Volinsky, Matrix factorization techniques for recommender systems, *Computer* **42**(8) (2009), 30–37.

[13] T. Geurts and F. Frasincar, *Addressing the cold user problem for model-based recommender systems*, in: Proceedings of the International Conference on Web Intelligence (WI 2017), ACM, 2017, pp. 745–752.

[14] P. Resnick and H.R. Varian, Recommender systems, *Communications of the ACM* **40**(3) (1997), 56–58.

[15] C. Stevens, *Knowledge-based assistance for accessing large, poorly structured information spaces*, PhD thesis, University of Colorado, 1993, `http://www.holodeck.com/curt/mypapers/Thesis.pdf`.

[16] R.J. Mooney, P.N. Bennett and L. Roy, *Book recommending using text categorization with extracted information*, in: Proceedings of the 3rd Association for the Advancement of Artificial Intelligence Workshop on Recommender Systems (AAAI 1998), AAAI, 1998.

[17] D. Jannach, P. Resnick, A. Tuzhilin and M. Zanker, Recommender systems: Beyond matrix completion, *Communications of the ACM* **59**(11) (2016), 94–102.

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

[19] A.I. Schein, A. Popescul, L.H. Ungar and D.M. Pennock, *Methods and metrics for cold-start recommendations*, in: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002), ACM, 2002, pp. 253–260.

[20] T. Westergren, *The music genome project*, 2007, `http://pandora.com/mgp`.

[21] X. Su and T.M. Khoshgoftaar, A survey of collaborative filtering techniques, *Advances in Artificial Intelligence* **2009** (2009), 4.

[22] R.D. Burke, Hybrid recommender systems: Survey and experiments, *User Modeling and User-Adapted Interaction* **12**(4) (2002), 331–370.

[23] C. Basu, H. Hirsh and W. Cohen, *Recommendation as classification: Using social and content-based information in recommendation*, in: Proceedings of the 15th National Conference on Association for the Advancement of Artificial Intelligence/Innovative Applications of Artificial Intelligence Conferences (AAAI/IAAI 1998), AAAI, 1998, pp. 714–720.

[24] D. Goldberg, D. Nichols, B.M. Oki and D. Terry, Using collaborative filtering to weave an information tapestry, *Communications of the ACM* **35**(12) (1992), 61–70.

[25] U. Shardanand and P. Maes, *Social information filtering: Algorithms for automating "word of mouth"*, in: Proceedings of the 13th Conference on Human Factors in Computing Systems (CHI 1995), ACM Press/Addison-Wesley Publishing Co., 1995, pp. 210–217.

[26] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom and J. Riedl, *GroupLens: An open architecture for collaborative filtering of netnews*, in: Proceedings of the 5th ACM Conference on Computer Supported Cooperative Work (CSCW 1994), ACM, 1994, pp. 175–186.

[27] J.S. Breese, D. Heckerman and C. Kadie, *Empirical analysis of predictive algorithms for collaborative filtering*, in: Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI 1998), Morgan Kaufmann Publishers Inc., 1998, pp. 43–52.

[28] A. Paterek, *Improving regularized singular value decomposition for collaborative filtering*, in: Proceedings of KDD Cup and Workshop (KDD 2007), Vol. 2007, ACM, 2007, pp. 5–8.

[29] R. Salakhutdinov and A. Mnih, *Bayesian probabilistic matrix factorization using Markov chain Monte Carlo*, in: Proceed-

ings of the 25th International Conference on Machine Learning (ICML 2008), ACM, 2008, pp. 880–887.

[30] D.D. Lee and H.S. Seung, *Algorithms for non-negative matrix factorization*, in: Proceedings of 14th Advances in Neural Information Processing Systems Conference (NIPS 2001), NIPS Foundation, 2001, pp. 556–562.

[31] M. Jamali and M. Ester, *A matrix factorization technique with trust propagation for recommendation in social networks*, in: Proceedings of the 4th ACM Conference on Recommender Systems (RecSys 2010), ACM, 2010, pp. 135–142.

[32] S.-T. Sun, E. Pospisil, I. Muslukhov, N. Dindar, K. Hawkey and K. Beznosov, *What makes users refuse web single sign-on?: An empirical investigation of OpenID*, in: Proceedings of the 7th Symposium on Usable Privacy and Security (SOUPS 2011), ACM, 2011, p. 4.

[33] N. Rubens, D. Kaplan and M. Sugiyama, *Active learning in recommender systems*, in: Recommender Systems Handbook, Springer, 2011, pp. 735–767.

[34] J. Nguyen and M. Zhu, Content-boosted matrix factorization techniques for recommender systems, *Statistical Analysis and Data Mining* **6**(4) (2013), 286–301.

[35] M. Elahi, F. Ricci and N. Rubens, *Active learning in collaborative filtering recommender systems*, in: Proceedings of the 15th International Conference on Electronic Commerce and Web Technologies (EC-Web 2014), Springer, 2014, pp. 113–124.

[36] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer and R. Harshman, Indexing by latent semantic analysis, *Journal of the American Society for Information Science* **41**(6) (1990), 391.

[37] D. Kim and B.-J. Yum, Collaborative filtering based on iterative principal component analysis, *Expert Systems with Applications* **28**(4) (2005), 823–830.

[38] R. Salakhutdinov, A. Mnih and G. Hinton, *Restricted Boltzmann machines for collaborative filtering*, in: Proceedings of the 24th International Conference on Machine Learning (ICML 2007), ACM, 2007, pp. 791–798.

[39] Y. Yu, C. Wang and Y. Gao, Attributes coupling based item enhanced matrix factorization technique for recommender systems, *CoRR* **abs/1405.0770** (2014).

[40] A. Kohrs and B. Merialdo, *Improving collaborative filtering for new-users by smart object selection*, in: Proceedings of the International Conference on Media Futures 2001 (ICMF 2001), 2001.

[41] D. Vandic, F. Frasincar and U. Kaymak, *Facet selection algorithms for Web product search*, in: Proceedings of the 22nd ACM International Conference on Information & Knowledge Management (CIKM 2013), ACM, 2013, pp. 2327–2332.

[42] A.M. Rashid, G. Karypis and J. Riedl, Learning preferences of new users in recommender systems: An information theoretic approach, *SIGKDD Explorations* **10**(2) (2008), 90–100.

[43] N. Golbandi, Y. Koren and R. Lempel, *On bootstrapping recommender systems*, in: Proceedings of the 19th ACM Conference on Information and Knowledge Management (CIKM 2010), 2010, ACM, 2010, pp. 1805–1808.