Chapter 23
On Bootstrapping Web Service Recommendation

Qi Yu

Abstract We present a novel framework to bootstrap Web Service recommendation. Service recommendation has become an effective means to achieve personalized service selection. It leverages past user-service interaction information to accurately predict user preference on previously unknown services. However, one key impediment has been the incompetence of current service recommendation systems in dealing with new users and services. Since a recommendation system has no knowledge about new users and services, it may completely fail to provide any recommendation or provide very poor ones. The proposed framework uses an agile interview process to quickly profile new users and services. The interview is structured by a decision tree that enables adaptive, intuitive, and rapid querying of users or services. We propose to exploit Non-negative Matrix Tri-Factorization (NMTF) to simultaneously partition users and services into a set of user and service groups. The group structure helps estimate the missing interaction information and also provides class labels to construct decision trees for both users and services, which will be used in the interview process. We conduct extensive experiments to assess the effectiveness of the proposed framework for bootstrapping service recommendation.

23.1 Introduction

Service Oriented Computing (SOC) offers an attractive paradigm for the provisioning and consuming of computing resources across a wide spectrum of domains. The large number of applications expected to heavily take advantage of SOC will lead to the deployment of substantial software services on the Web. Many Web services may also offer similar functionalities but vary from each other in terms of the Quality of Service (QoS) they deliver [16]. The QoS is mainly made of user centered quality

Q. Yu (✉)
Rochester Institute of Technology, Rochester, USA
e-mail: qi.yu@rit.edu

parameters and examples include availability, response time, throughput, and so on. As the number of Web services is expected to grow far beyond the reach of any manual effort, a key challenge is to automatically assess the QoS of large-scale Web services. This will enable casual service users to easily select the Web service that best fulfills their QoS requirements [17].

The distributed and dynamic SOC environment leads to very diverse QoS experience for service users. Users may locate in different network environments and have different physical distances with the Web services they access. These discrepancies imply that different users may receive significantly different QoS from the same Web service. Service recommendation systems explicitly consider user discrepancies by leveraging a Collaborative Filtering (CF) scheme [5, 11, 15, 18, 19]. CF assumes that users who have common QoS experiences with some services may share similar experiences with other services. It exploits similar users' QoS experience to accurately predict the QoS that an active user may receive from previously unknown services. In this way, personalized service selection can be achieved that enables users to conveniently choose the most suitable services from a large number of previously unknown candidates.

Service recommendation systems rely on the historical user-service interaction to make QoS predictions. The similarity between two users is measured based on the QoS of their commonly invoked services. Similarly, the similarity between two services is evaluated by the QoS received by the set of users that invoke both services. Sufficient historical QoS information increases the chance to identify similar users or services, which is central to the effectiveness of CF based recommendation systems. Therefore, the more knowledge the system has on the users and services, the more accurate QoS prediction can it provide. Existing service recommendation systems perform reasonably well on warm-start users for which they possess adequate information. One key impediment has been their incompetence in dealing with new users and services, which is usually referred to as the cold start issue. As the system possesses very little or no historical QoS information from new users and services, it may fail to provide any recommendation or provide very poor ones.

Due to the wide adoption of SOC in both industry and government, new services are being increasingly deployed and new users keep entering the SOC market. Hence, effectively dealing with the cold-start issue is critical in attracting new users and service providers, which is instrumental for SOC to reach its full potential. An initial interview process is commonly used to elicit user's information in many recommendation systems. User profiles are constructed based on the interview results, which will then be used for recommendation. The initial interview process should be both short and intuitive so that a new user won't get bored or lost. Another desirable feature of the interview process is to adaptively query the user based on the results of prior interview questions [3, 9].

Decision trees have been employed to conduct initial interviews to bootstrap e-commerce recommendation systems (e.g., Amazon and Netflix) [4, 21]. A ternary tree is recursively constructed by selecting an item to split existing users assigned to a tree node into its three child nodes along branches, labeled as "like", "dislike", and "unknown", respectively. During the interview, the new user is expected to rate an item chosen by the decision tree at each given step. Based on the rating, she will be directed to one of the child nodes. The interview continues until the new user is assigned to a leaf node, which represents a homogeneous group of existing users. These users are considered to be as the similar users and will be used to predict the new user's preferences on different items.

Bootstrapping service recommendation poses some new challenges, which hinder a direct application of ternary decision trees. In e-commerce recommendation systems, users' preferences on items are typically represented by few categorical rating values, (e.g., 1–5). This enables a straightforward way to assign users into different groups based on their ratings on an item. For example, the ternary tree assigns a user into the "like" group if her rating is no less than a predefined threshold (e.g., 3) and "dislike" group if otherwise. In contrast, the QoS data used in service recommendation is described by continuous attributes (e.g., 1.2 s response time and 0.95 availability). Therefore, dividing users into groups is less intuitive and demands some principled criterion.

Dealing with incomplete QoS data gives rise to the second key challenge. Since an existing user may only invoke a limited number of services, only a small subset of QoS data is observed. The ternary tree introduces the "unknown" tree node to group together users with no ratings on the selected item. Users in this node share a similar opinion on the item, which may be interpreted as either "not know" or "no interest". Correspondingly, no response is also allowed during the interview process, which directs the new user into the "unknown" node. When bootstrapping service recommendation, a new user is expected to invoke a small number of selected services during the interview.[1] Invoking a service always generates a response. Even when the service is down, a "time out" message is returned, indicating that the service is not available. This is different from rating an item (e.g., a product or a movie), which may result in no response when the user does not know or has no interest in the item. In contrast, no response is no longer an option during the interview for service recommendation. If a ternary tree is used, the "unknown" nodes will never be visited during the interview. This in essence ignores a large number of users that are assigned into these nodes and hence dramatically reduces the chance to locate similar users.

We develop a novel framework to bootstrap Web service recommendation. In a preliminary version of this paper, we developed a strategy to provide high-quality service recommendations for new service users [14]. One key extension of the proposed framework has been the ability to deal with new services. In this way, it provides a complete solution to the cold-start issue in service recommendation by tackling both new users and services. In particular, we propose to exploit Non-negative Matrix Tri-Factorization (NMTF) to simultaneously partition users and services into a set of user and service groups. The group structure helps estimate the missing interaction information and also provides class labels to construct decision trees for both users and services. The decision tree is used to structure an initial interview that

[1] The service invocation code can be wrapped as a small software toolkit that is easily accessed by end users.

enables adaptive, intuitive, and rapid querying of users or services. At the end of the interview, a new user or service will be classified into one of the user or service groups obtained by NMTF. Since the group is deemed to be comprised of users or services that are similar to the new user or service, the QoS of the new user or service can be predicted based on the QoS of its similar users or services. One challenging issue that is particular to profiling new services is that the interview process requires to query existing users. However, a selected user may not want to participate in the interview process. The user groups resulted from NMTF enable us to choose alternative users that are similar to the selected users to work as surrogates in the decision tree. Experimental results clearly demonstrate the effectiveness of the surrogate user strategy in profiling new services.

The remainder of the paper is organized as follows. We review some existing works that are most relevant to ours in Sect. 23.2. We describe in detail the proposed framework for bootstrapping service recommendation systems in Sect. 23.3. We assess the effectiveness of the proposed framework via a set of experiments in Sect. 23.4. We conclude in Sect. 23.5.

23.2 Related Work

The ever increasing number of Web services demands systematic approaches to facilitate service users in efficiently and accurately retrieving services that match both their functional and non-functional requirements. Collaborative Filtering (CF) based techniques have been recently adopted to provide personalized service recommendation to users [5, 11, 18, 19]. Shao et. al. present a service recommendation system by assuming that similar users tend to receive similar QoS from similar services [11]. This is in essence a standard user-based CF algorithm. Zheng et. al. enhance the user-based approach by integrating item-based CF, which results in a hybrid algorithm with better prediction accuracy [19]. Complementary information, such as users' locations [1, 2], invocation frequencies of services [10], and query histories of users [18], has also been leveraged to improve the quality of recommendation.

Both user- and item-based approaches follow the neighborhood centric strategy in CF, which explores the local neighborhood to identify similar users or items for recommendation. Zheng et. al. recently proposed a model based CF algorithm that achieves higher prediction accuracy [20]. The proposed algorithm uses the user-based approach as a precursor to identify top-k similar users. Based on the user neighborhood information, matrix factorization is employed to construct a global model, which can be used to predict unobserved QoS data. Different from our strategy, an unconstrained version of matrix factorization is used, which does not discover the user groups.

All existing service recommendation approaches focus on predicting QoS for warm-start users. To our best knowledge, there is no existing work that provides a systematic approach for cold-start service recommendation. Dealing with cold-start users has received considerable attention in e-commerce recommendation systems.

An initial interview has been suggested as an effective way to quickly build new users' profiles. Based on how the seed questions are selected, there are two types of interviews. The first type of interview chooses a static seed set based on some principled criteria, such as coverage [3], popularity [8], and discriminative power [9]. Users always answer a fixed set of questions, which does not fully leverage the interactive nature of the interview process. Recent works propose to adaptively query users based on their responses to the prior interview questions [9, 4, 21]. Decision trees appear as an ideal vehicle to carry out the adaptive initial interview. A ternary tree suits perfectly for the rating-based recommendation systems, which are commonly used in e-commerce. As discussed in Sect. 23.1, service recommendation poses some new challenges that make a ternary tree inapplicable. In particular, since the "unknown" nodes in a ternary tree will never be visited during the interview, valuable information carried by existing users cannot be fully leveraged to provide high quality recommendations.

23.3 Cold-start Service Recommendation

We present the framework for bootstrapping service recommendation in this section. The proposed framework simultaneously deals with both new users and services. It exploits Non-negative Matrix Tri-Factorization (NMTF) to discover the user and service group structures from a set of incomplete QoS data that captures the historical user-service interactions. The tree learning algorithm then constructs two decision trees to partition the users and services, respectively, to fit the group structure discovered by NMTF. The simple structure and interpretability of the decision tree serve ideally for an initial interview process, which adaptively queries new users and services for rapid profiling.

23.3.1 NMTF for User and Service Group Discovery

Before delving into the technical details, we first describe the symbols and notations that are used throughout the paper. Assume that there are n existing users and m Web services. The QoS attribute (e.g., response time, reliability, and availability) under consideration takes positive real values. We use a matrix $A \in \mathbb{R}_{+}^{m\times n}$ to denote the QoS data, where A_{ij} represents the QoS that service i delivered to user j. In this regard, the i-th row of A represents service s_i while the j-th column of A represents user u_j. This essentially models user u_j as an m-dimensional feature vector, in which each element u_{jq} signifies u_j's interaction with s_q.

NMTF computes three low-rank matrices, the service coefficient matrix $F \in \mathbb{R}_{+}^{m\times k}$, the prototype matrix $B \in \mathbb{R}_{+}^{k\times l}$, and the service coefficient matrix $G \in \mathbb{R}_{+}^{n\times l}$ to approximate the original QoS matrix A, i.e., $A \approx FBG'$. In particular, the $k \times l$ prototype matrix B is deemed to provide a compact representation for the original QoS matrix A with a $k \times l$ block structure. In this regard, the columns of B, $\{b_1, ..., b_l\}$, correspond to the l different types of users and the rows of B, $\{b'_1, ..., b'_k\}$, correspond to the k different types of services.

Let $v_q = Fb_q$ denote the q-th column vector of $V = FB$, where $q = 1, ..., l$. The m dimensional vector $v_q \in \mathbb{R}^m$ reflects how each service interacts with the q-th type of users. Therefore, the columns of V are considered to form a new basis, where each basis vector captures the QoS related latent feature of one (out of l) type of users. Consequently, the user coefficient matrix G is the new representation of the users under this new basis. G can also be regarded as a projection of A onto the latent user feature space V. More specifically,

$$a_j \approx \sum_{q=1}^{l} G_{jq} v_q \tag{23.1}$$

where a_j is the j-th column vector in A, representing user u_j. Equation (23.1) shows that each user vector a_j is approximated by a linear combination of the column vectors in V weighted by the components of G. Similarly, the service coefficient matrix F is a projection of A' onto the latent service feature space $R' = GB'$,[2] i.e., $a'_i \approx \sum_{p=1}^{k} F_{ip} r'_p$.

The latent user feature space V together with the new representation matrix G should provide a good approximation of the original QoS data matrix A. Since only a small subset of QoS data is observed, we introduce a weight matrix W, where $W_{ij} = 1$ if A_{ij} is observable and $W_{ij} = 0$ otherwise. Therefore, we compute F, B, and G by solving the following optimization problem:

$$\min_{F \geq 0, G \geq 0} J = \left\| W; (A - FBG') \right\|^2 \tag{23.2}$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{n} W_{ij} \left(A_{ij} - \left(FBG' \right)_{ij} \right)^2 \tag{23.3}$$

where ; is component-wise matrix product and $|| \cdot ||$ is matrix norm. Since all the components of A take non-negative values, we also enforce a non-negative constraint on matrices F, B, and G. As can be seen from Eq. (23.1), the nonnegative constraint ensures that a user vector is an additive linear combination of the new basis vectors. This allows a more intuitive interpretation than other matrix factorization approaches, such as Singular Value Decomposition (SVD), where negative values are allowed in the matrix components.

[2] Along the same lines, a'_i, the i-th row vector in A, representing service s_i, is approximated by a linear combination of the row vectors in BG' weighted by the components of F: $a'_i \approx \sum_{p=1}^{k} F_{ip} r'_p$. $r'_p = b'_p G'$ is the p-th row of BG', which reflects how each user interacts with the p-th type of services

23.3.2 Decision Tree Learning for User and Service Profiling

Due to its simplicity, interpretability, and the ability to adaptively query users, decision tree becomes an ideal tool to perform the initial interview via which a new user's profile can be constructed. As motivated in Sect. 23.1, the continuous nature of the QoS attributes and the limited observable QoS data pose key challenges to build a decision tree. The latent feature space discovered via matrix factorization carries rich information that is instrumental to understand the interaction patterns between users and services. It plays a critical role in learning a decision tree from a set of incomplete QoS data. More specifically, the latent feature space enables us to:

- discover homogeneous user and service groups that contain similar users and services;
- estimate the unobserved entries in the QoS matrix A.

Since the matrix G (or F) is a projection onto the latent user (or service) feature space, it naturally captures the user (or service) group structure. More intuitively, users (or services) that share similar latent features should have similar representations in the latent feature space. To make sure that each user (or service) is assigned to only one user (or service) group (i.e., hard group membership), we enforce constraints $GG' = \text{diag}(|\mathcal{U}_1|, ..., |\mathcal{U}_l|)$ and $FF' = \text{diag}(|\mathcal{S}_1|, ..., |\mathcal{S}_k|)$. This makes G (or F) a user (or service) group indicator matrix:

$$G_{jq} = \begin{cases} 1 \text{ if } u_j \in \mathcal{U}_q \\ 0 \text{ otherwise} \end{cases} \tag{23.4}$$

$$F_{ip} = \begin{cases} 1 \text{ if } s_i \in \mathcal{S}_p \\ 0 \text{ otherwise} \end{cases} \tag{23.5}$$

where $\mathcal{U}_q$ is the q-th user group and $|\mathcal{U}_q|$ denotes the number of users assigned to the group. Similarly, $\mathcal{S}_p$ is the p-th service group and $|\mathcal{S}_p|$ denotes the number of services assigned to the group. These constraints ensure that each row of G (or F) has only one non-zero element, which denotes the group that the user (or service) is assigned to.

The second key usage of the latent feature space is to estimate the missing QoS entries. If the latent features indeed capture the interaction patterns between users and services, they are expected to provide a good estimation of the unobserved QoS data. More specifically, the QoS that an unknown service s_i will deliver to a user u_j can be estimated as:

$$A_{ij} \approx \hat{A}_{ij} = \sum_{p=1}^{k} \sum_{q=1}^{k} V_{ip} G_{jq} \tag{23.6}$$

The j-th column vector of the completed matrix $\hat{A}$ corresponds to user u_j and the j-th row vector of matrix G encodes the class (or group) label for the user. The class labels from G allow us to exploit the classical information gain as the principled criterion to select services to be used as the tree nodes. Using the completed matrix $\hat{A}$ avoids the generation of "unknown" nodes, which are never visited during the initial interview for service recommendation. Instead of a ternary tree, our tree learning algorithm generates a binary decision tree, via which all existing user information can be leveraged to construct a new user's profile. To ensure a concise interview process, we employ two strategies to control the depth of the tree. First, we stop splitting the a node if the number users assigned to it is less than a predefined threshold value. Second, we exploit a standard pruning process to merge and join leaf nodes after the tree is fully grown.

Figure 23.1 shows an example decision tree constructed from a real-world QoS dataset obtained from [19]. Each internal node of the tree represents a service. Based on the QoS value, users are directed to one of its child nodes. For example, if the response time that a user received from service s_{53} is less than 0.74 s, she will be directed to child node s_{59}. At this node, the response time of the user will be evaluated against the service in the node. This process continues until the user reaches one of the leaf nodes, which corresponds to one of the user groups.

In what follows, we present an important property of the binary decision tree as constructed by following the above procedure. This helps justify why it can provide high-quality service recommendations for cold-start users and services.

Theorem 23.1 *A decision tree that exploits class labels provided by matrix G partitions users into cohesive user groups, where G is computed by minimizing objective function J with constraint in Eq. (23.4).*

Proof Since each column vector of A corresponds to a user, we reformulate the objective function J using column vectors of A.

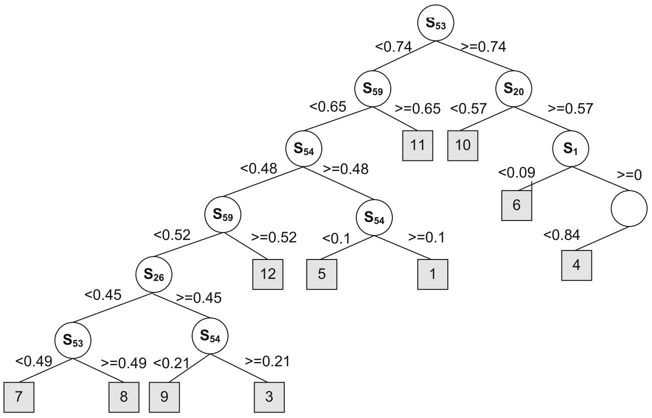


Fig. 23.1 An example decision tree for new user interview

$$J = \sum_{j=1}^{n} \left\| w_j ; [a_j - \sum_{q=1}^{k} G_{jq} v_q] \right\|^2 \tag{23.7}$$

$$= \sum_{j=1}^{n} \left\| \sum_{q=1}^{k} G_{jq} [w_j ; (a_j - v_q)] \right\|^2 \tag{23.8}$$

$$= \sum_{j=1}^{n} \sum_{q=1}^{k} G_{jq} \left\| w_j ; (a_j - v_q) \right\|^2 \tag{23.9}$$

$$= \sum_{q=1}^{k} \sum_{u_j \in \mathcal{U}_q} \left\| w_j ; (a_j - v_q) \right\|^2 \tag{23.10}$$

where w_j is the j-th column of W and $\circ$ is element-wise vector product. Due to Eq. (23.4) and the fact that one user is assigned to only one group, we have $\sum_{q=1}^{k} G_{jq} = 1$, which leads Eqs. (23.7) – (23.8). From Eqs. (23.8) – (23.9), we use the fact $G_{jq}^2 = G_{jq}$ since $G_{jq} = 1$ or 0. Finally, $G_{jq} = 1$ only when $u_j \in \mathcal{U}_q$ gives Eq. (23.10).

Minimizing objective function J is equivalent to find the optimal set $\{(\mathcal{U}_q, v_q) | q \in (1, k)\}$ that minimizes Eq. (23.10). We know that $\mathcal{U}_q$ denotes the q-th user group and the group membership is encoded by G. If we can find out what the latent feature vector v_d denotes, we are able to interpret what matrix factorization actually achieves under constraint specified in Eq. (23.4). Since the optimal V minimizes J, in order to find out V, we take the partial derivative of J with respect to V:

$$\frac{\partial J}{\partial V} = -2(W ; A)G + 2(W ; (VG'))G \tag{23.11}$$

$$= -2(W ; (-A + VG'))G \tag{23.12}$$

Setting $\frac{\partial J}{\partial V} = 0$ gives $-A + VG' = 0$. Multiplying both sides by G gives $VG'G = AG$. Using the fact $GG' = \mathrm{diag}(|\mathcal{U}_1|, ..., |\mathcal{U}_k|)$, we get

$$|\mathcal{U}_q| v_q = \sum_{j=1}^{n} G_{jq} a_j \tag{23.13}$$

$$v_q = \frac{1}{|\mathcal{U}_q|} \sum_{j=1}^{n} G_{jq} a_j \tag{23.14}$$

$$= \frac{1}{|\mathcal{U}_q|} \sum_{u_j \in \mathcal{U}_q} G_{jq} a_j \tag{23.15}$$

We exploit Eq. (23.4) in the last step of derivation.

Equation (23.13) reveals that v_q is actually the centroid of the q-th user group. Hence, we conclude that minimizing J with constraint in (23.4) is equivalent to performing k-means clustering on the existing users. The result is a set of cohesive user groups with minimal total squared deviation from their group means (or centroids). As G encodes the group memberships, our tree learning algorithm aims to construct a decision tree that partitions users into the same set of cohesive user groups.

Theorem 23.2 *A decision tree that exploits class labels provided by matrix F partitions services into cohesive service groups, where F is computed by minimizing objective function J with constraint in Eq. (23.5).*

23.3.2.1 Profiling New Users and Services

Profiling a new user is straightforward by following an initial interview structured by a decision tree like the one in Fig. 23.1. During the interview, the user invokes the services on the tree nodes until being directed into one of the leaf nodes, which represents a user group. The new user hence is expected to share similar QoS experience with other users in the same group. To make the interview process painless, the service invocation code can be wrapped as a small software toolkit that is easily accessed by end users.

Profiling a new service, on the other hand, is a little bit more complicated. In the decision tree for new service interview, each internal tree node represents a user. During the interview process, the users on the tree nodes need to invoke the new service and report their QoS. Since a number of users are involved in the interview process, it will take longer than interviewing new users, which just invokes a set of services. In fact, rapidness of the interview process is not critical for profiling new services as it makes sense that introducing a new service into the market may take some time. However, the key issue is that there might be some users that do not want to participate in the interview. One possible solution is to adopt some bonus mechanism to stimulate users. In addition, we propose to use a surrogate user strategy to improve the response rate of service users. The surrogate user strategy benefits from the co-clustering nature of NMTF, which simultaneously clusters both users and services. While the goal of new service profiling is to classify a new service into one of the service groups obtained by NMTF, the user groups provide options to choose alternative users when a selected user is not willing to participate in the interview. More specifically, when a user u_i fails to provide QoS information on the new service s_t, we randomly choose another user u_j from user group $\mathcal{U}_q$, where $u_i \in \mathcal{U}_q$, to replace u_i. Since u_j and u_i are from the same user group, they are expected to receive similar QoS from s_t. Therefore, it is highly probable that s_t will be directed to the same path in the decision tree when u_j is queried instead of u_i. This will have the effect of leading s_t to the same service group as when u_i is queried during the interview. Our experimental results in Sect. 23.4 demonstrate the effectiveness of the surrogate user strategy.

23.3.3 Computing G, B, and F

Matrices G, B and F play key roles in both decision tree learning and cold-start service recommendation. G, B and F can be derived by solving the optimization problem in Eq. (23.2). However, minimizing J under constraints specified by Eq. (23.4) and Eq. (23.5) is non-trivial. Since there is no analytical solution for that, we develop an iterative algorithm to efficiently compute G, B and F.

The constraints in Eq. (23.4) and Eq. (23.5) require binary values on the components of G and F, which makes the optimization problem unsolvable [12]. To resolve this issue, we instead enforce the following constraints: $G1 = 1$ and $F1 = 1$. This is equivalent to enforcing a soft group membership. Take the user group as an example and the same idea is applied to the service group. From $G1 = 1$, we have $\sum_{q=1}^{k} G_{jq} = 1, \forall j \in [1, n]$. Hence, G_{jq} can be interpreted as the probability that u_j belongs to group $\mathcal{U}_q$. User u_j will be assigned to group $\mathcal{U}_{\hat{q}}$, where

$$\hat{q} = \arg\max_{q}\{G_{jq} | 1 \leq q \leq k\}$$

We incorporate these new constraints into objective function J as penalty terms, which lead to the following objective function:

$$\min_{F \geq 0, B \geq 0, G \geq 0} J(G, B, F) = \left\| W; (A - FBG') \right\|^2 + \alpha ||G1 - 1||^2 + \beta ||F1 - 1||^2 \tag{23.16}$$

In order to minimize $J(G, B, F)$, the proposed iterative algorithm updates G, B and F alternatively. That is, while $J(G, B, F)$ is minimized with respect to G, B and F will be fixed and vice versa. The update of G, B and F is performed by using a set of update rules, which guarantee the convergence of the iterative algorithm. The update rules are derived based on a set of auxiliary functions of objective function $J(G, B, F)$, which are formally defined as follows.

Definition 23.1 $Z(G, \tilde{G})$ is an auxiliary function of function $J(G)$ if it satisfies the following conditions for any G and $\tilde{G}$*:* $Z(G, \tilde{G}) \geq J(G)$*;* $Z(G, G) = J(G)$ *[6].*

Now, let's plug the auxiliary function into our iterative algorithm and see how we can exploit it to derive the update rules. Let $J(G)$ denote the part of $J(G, F)$ that is only relevant to G. Assume that $\{G^{(1)}, ..., G^{(t)}, ...\}$ is a set of matrices obtained by the iterative algorithm, where (t) denotes the t-th iteration. Assume that G is updated using the following update rule:

$$G^{(t+1)} = \arg\min_{G} Z(G, G^{(t)}) \tag{23.17}$$

where $G^{(t)}$ and $G^{(t+1)}$ are matrix G at the t-th and $(t + 1)$-th iterations, respectively. It is straightforward to show that $J(G)$ monotonically decreases under update rule in Eq. (23.17):

$$J(G^{(t)}) = Z(G^{(t)}, G^{(t)}) \geq Z(G^{(t)}, G^{(t+1)}) \geq J(G^{(t+1)})$$

Following the same lines, we can use similar update rules for B and F. Since the iterative algorithm updates G, B and F in turn, we have

$$\begin{aligned} J(F^{(t)}, B^{(t)}, G^{(t)}) &\geq J(F^{(t+1)}, B^{(t)}, G^{(t)}) \geq J(F^{(t+1)}, B^{(t+1)}, G^{(t)}) \\ &\geq J(F^{(t+1)}, B^{(t+1)}, G^{(t+1)}) \end{aligned}$$

As $J(G, B, F)$ is apparently lower bounded, it is guaranteed to converge under the above update rules. What remains is to derive the update rules, which requires to find suitable auxiliary functions for $J(G, B, F)$ and compute their global minima.

Theorem 23.3 *Let*

$$J(G) = \left\| W; (A - FBG') \right\|^2 + \alpha ||G1 - 1||^2 \tag{23.18}$$

The auxiliary function of $J(G)$ is given by

$$Z(G, \tilde{G}) = Z_1(G, \tilde{G}) + Z_2(G, \tilde{G}) \tag{23.19}$$

where,

$$\begin{aligned} Z_1(G, \tilde{G}) = \sum_{ij} W_{ij} \Bigg[& A_{ij}^2 - 2 \sum_{pq} A_{ij} F_{ip} B_{pq} \tilde{G}_{jq} \left(1 + \log \frac{G_{jq}}{\tilde{G} jq}\right) \\ & + \sum_{pq} [FB\tilde{G}']_{ij} F_{ip} B_{pq} \frac{G_{jq}^2}{\tilde{G}_{jq}} \Bigg] \end{aligned} \tag{23.20}$$

$$\begin{aligned} Z_2(G, \tilde{G}) = \alpha & \sum_{jq} \left([\tilde{G}1]_j \frac{G_{jq}^2}{\tilde{G}_{jq}} \right) \\ & - \alpha \sum_{jq} 2\tilde{G}_{jq} \left(1 + \log \frac{G_{jq}}{\tilde{G}_{jq}}\right) + n\alpha \end{aligned} \tag{23.21}$$

The global minimum of $Z(G, \tilde{G})$ is

$$G_{jq} = \tilde{G}_{jq} \left[\frac{[(W; A)'FB]_{jq} + \alpha}{[(W; (FB\tilde{G}'))'FB + \alpha \tilde{G} E]_{jq}} \right]^{\frac{1}{2}} \tag{23.22}$$

Proof Sketch It is straightforward to show that $Z(G, G) = J(G)$. Furthermore, $J(G)$ has two quadratic terms. Applying Jensen's inequality and inequality $x \geq 1 + \log x, \forall x > 0$ when expanding both terms, we get

$$\left\| W; (A - FBG') \right\|^2 \leq Z_1(G, \tilde{G}) \tag{23.23}$$

$$\alpha||G1 - 1||^2 \leq Z_2(G, \tilde{G}) \tag{23.24}$$

From Eqs. (23.23) and (23.24), we have $Z(G, \tilde{G}) \geq J(G)$. Hence, we prove that $Z(G, \tilde{G})$ is an auxiliary function of $J(G)$.

To show that Eq. (23.22) gives the global minimum of $Z(G, \tilde{G})$, we need to first prove that $Z(G, \tilde{G})$ indeed has a global minimum. This can be achieved by showing that $Z(G, \tilde{G})$ is a convex function on G. We compute the second order derivative of $Z(G, \tilde{G})$ with respect to G, which gives the Hessian matrix of $Z(G, \tilde{G})$:

$$\frac{\partial^2 Z(G, \tilde{G})}{\partial G_{jq} \partial G_{pr}} = \delta_{jp}\delta_{qr} \left(\frac{2[W; AFB]_{jq}\tilde{G}_{jq} + 2\alpha\tilde{G}_{jq}}{G_{jq}^2} + \frac{2[W; (FB\tilde{G}')FB]_{jq} + 2\alpha[\tilde{G}E]_{jq}}{\tilde{G}_{jq}} \right)$$

where $\delta_{ab} = 1$ when $a = b$ and 0 otherwise. Hence, the Hessian is a diagonal matrix with positive diagonal elements, which makes it positive definite. Therefore, $Z(G, \tilde{G})$ is a convex function on G. To compute the global minimum, it is sufficient to compute its local minimum. We set $\frac{\partial Z(G, \tilde{G})}{\partial G_{jq}} = 0$ and through some algebra, we get Eq. (23.22).

Following the same lines, we can derive the update rules for F and B:

$$F_{iq} = \tilde{F}_{iq} \left[\frac{[W; ABG']_{ip} + \beta}{[W; (\tilde{F}BG')BG' + \beta\tilde{F}E]_{ip}} \right]^{\frac{1}{2}} \tag{23.25}$$

$$B_{pq} = \tilde{B}_{pq} \left[\frac{[F'(W; A)G]_{pq}}{[F'(W; (F\tilde{B}G'))G]_{pq}} \right]^{\frac{1}{2}} \tag{23.26}$$

Having update rules (23.22), (23.25), and (23.26), the iterative algorithm essentially updates F, B and G alternatively in each iteration. The algorithm continues until it converges or a predefined number of iterations is reached.

23.4 Experiments

We carry out a set of experiments to evaluate the effectiveness of the proposed framework for boostrapping service recommendation. The experiments are conduced on a real-world QoS dataset that consists of 1.5 million service invocation records. 150 computer nodes from the Planet-Lab,[3] which are located in over twenty countries, are

[3] http://www.planet-lab.org

leveraged to automatically invoke a hundred selected Web services. These services are distributed across more than twenty countries. Each computer node invokes each service for 100 times and the average Round-Trip Time (RTT) is used as the QoS dataset in our experiments.

23.4.1 Experiment Design

We organize the QoS data into a 100×150 matrix A, in which entry A_{ij} denotes the averaged RTT that user j used to invoke service i. We randomly remove a certain percentage (80–96 %) of entries from A to simulate a real-world QoS dataset, where only a small subset of entries are observed. To assess the proposed bootstrapping strategy, we follow a similar design as in [21], which splits users (or services) into two disjoint subsets: the training set and the test set, consisting of 80 and 20 % users (or services), respectively. We apply NMTF to the training set to discover the user and service groups and estimate the missing QoS entires. We then construct the decision tree for the initial interview. The actual RTT records of the test users are used to simulate the results of invoking the services in the decision tree.

We employ Mean Absolute Error (MAE), one of the most widely used metric in recommendation systems, to assess the quality of recommendation:

$$MAE = \sum_{i,j} \frac{|A_{ij} - \hat{A}_{ij}|}{N} \tag{23.27}$$

where A_{ij} and $\hat{A}_{ij}$ denote the actual and estimated RTT respectively. N is the total number of estimated QoS entries. Since the RTT entries are randomly removed, all the results reported below are obtained by computing the average over 20 runs. The numbers of user and service groups are 30 and 20, respectively. The default value for the penalty terms α and β are both set to 10. These default values will be used in all the experiments unless specified otherwise.

23.4.2 Quality of Cold-start Recommendation

To our best knowledge, there is no existing work on providing service recommendation for cold-start users. As discussed in Sect. 23.1, the ternary tree approach presented in [4, 21] is not suitable for the initial interview of service recommendation, either. To demonstrate the effectiveness of the proposed bootstrapping strategy, we implemented four representative collaborative filtering methods, including:

- the user based algorithms using both Pearson Correlation Coefficient (UPCC) and cosine similarity (referred to as UCOS) as similarity measures [11];

- the item based algorithm using Pearson Correlation Coefficient (IPCC) as similarity measure;
- the hybrid collaborative algorithm that combines both user and item based approaches using their prediction accuracy as the aggregation weights (referred to as WSRec) [19].
- the constrained matrix factorization model (referred to as NMTF), as discussed in Sect. 23.3, in which Eq. (23.6) is utilized to make the prediction after the model is constructed.

We apply the above methods to the warm-start users and use the obtained result as the baseline to assess our cold-start performance. As indicated in [21], using a ternary tree model, the cold-start performance is always worse than the warm-start performance considering that more information is available for the warm-start users. Therefore, the relative warm/cold start performance is a good indicator about the effectiveness of the bootstrapping process.

Figure 23.2 compares the warm-start MAE performance from the four representative CF algorithms with the MAE performance for cold-start users from the proposed bootstrapping framework (referred to as NMTF-DT). We vary the sparsity ratio of the QoS matrix A from 80 to 96 % and achieve two important observations. First, the cold-start performance of NMTF-DT outperforms the warm-start performance of other algorithms in all cases. This clearly demonstrates the effectiveness of the bootstrapping strategy. As can be seen later in Fig. 23.6, a new user only needs to invoke few services (3–6 on average) during the interview process. This is usually much smaller than the number of services invoked by a warm-start user. For example, if the sparsity of A is 80 %, since we have 100 services in total, each existing user invoked 20 services on average. The fundamental reason for this is that the integration of MF with decision tree learning identifies the most important few services to invoke for the new user. The QoS collected from these services captures the key

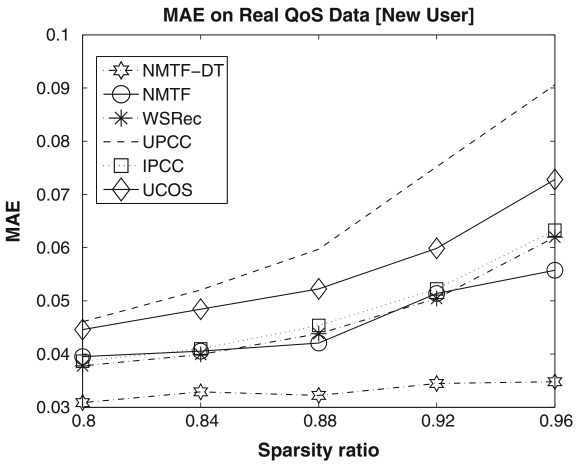


Fig. 23.2 Cold-start user MAE performance comparison

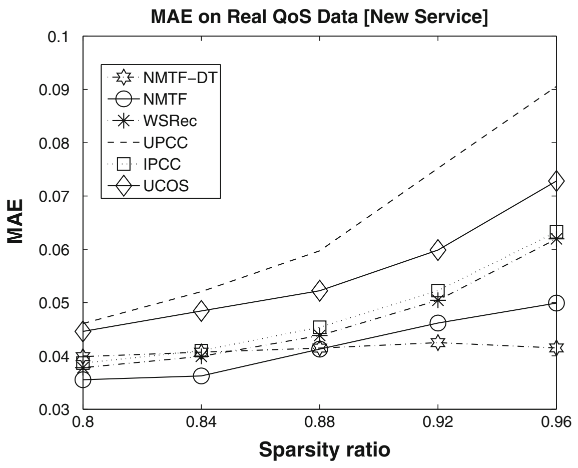
(latent) features of the user. This is critical to discover the most similar user group, which is used to predict the QoS from other services that are unknown to the new user.

Second, as the sparsity of A increases, the performance advantage of NMTF-DT becomes more significant. This is because the warm-start performance drops quickly when less information is available for users. For a very sparse QoS dataset, most users may invoke very few or even zero services. In fact, these algorithms essentially suffer from the cold-start problem, which we aim to resolve in this paper. The MAE performance of NMTF-DF also drops as sparsity increases because it relies on the similar user groups to make the prediction. However, the performance goes down much slower than other algorithms. It is also interesting to note that NMTF suffers less than other algorithms for the cold-start issue. This also contributes to the good performance of NMTF-DT, in which NMTF serves as a precursor of the entire bootstrapping process.

Figure 23.3 shows the results on cold-start services. In this set of experiments, we randomly choose 80 % rows from matrix A, which represent 80 % of the services, and use these services as the training set. The remaining rows are used as the test set. The results show a very similar trend as in Fig. 23.2. The MAE performance on cold-start services is a little bit worse than the performance on cold-start users but it is still comparable with the warm-start performance achieved by NMTF. Furthermore, it outperforms NMTF when the data becomes very sparse.

Figure 23.4 demonstrates the effectiveness of the proposed surrogate user mechanism for profiling new services. In this set of experiments, we randomly choose a user from the query path and replace it with a user that is randomly chosen from the user group where the original user belongs to. As can be seen, using the surrogate user delivers a MAE performance, which is almost identical to using the original user. It is also interesting to see that using the surrogate user sometimes even achieves better MAE performance. This may be due to some noises that affect the QoS delivery

Fig. 23.3 Cold-start service MAE performance comparison



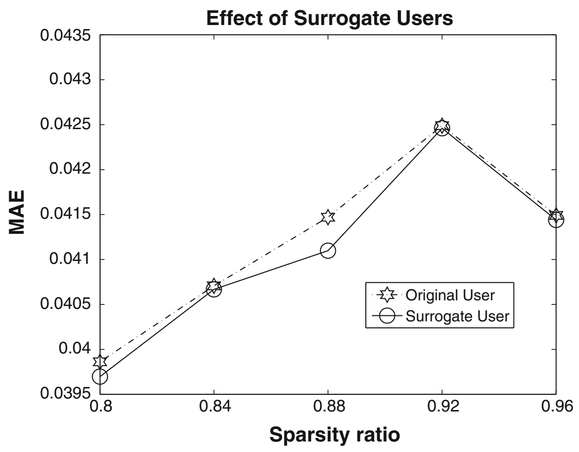


Fig. 23.4 Effectiveness of surrogate users

to the original user. The surrogate user may not be affected by these noises, which contributes to a better estimation of the QoS of the new service.

23.4.3 Impact of Parameters

We investigate the impact of two important parameters in this section, including the height of the decision tree and the number of user groups. The sparsity ratio of A is kept as 80 %.

We control the height of the decision tree by restricting the minimum number of services per leaf node, referred to as min_ leaf_ size. As shown in the left chart of Fig. 23.5, when we vary min_ leaf_ size from 1 to 10, the average tree height decreases from 13.36 to 6.16. The MAE performance on cold-start users also decreases slowly as tree height decreases although there are some small fluctuations

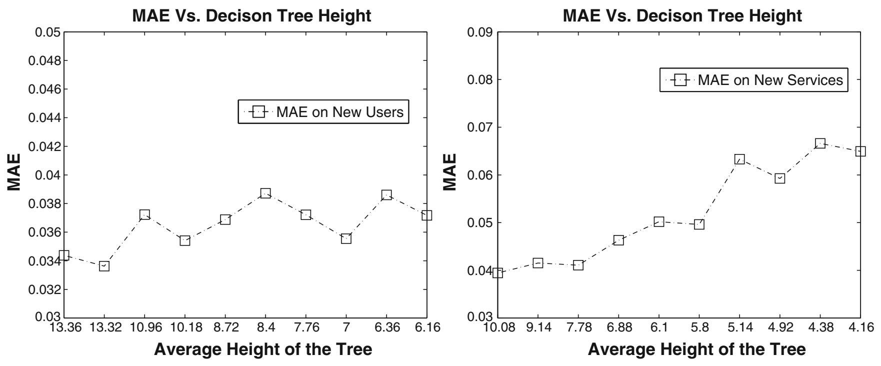


Fig. 23.5 Impact of decision tree height

due to the randomness in removing entries from A and the initialization of F, B and G. A very similar trend is shown in the right chart of the figure, which gives the result on cold-start services.

A service (or user) may appear multiple times in the decision tree. For example, in Fig. 23.1, services S_{53}, S_{59} and S_{54} all appear more than one times in the example decision tree. Therefore, the number of services (or users) that need to be queried by a new user (or service) during the interview is actually much smaller than the tree height. Figure 23.6 reports the average number of service invocations versus the min_ leaf_ size, which confirms our hypothesis. It is obvious only very small number (3–6 on average) of services (or users) need to be queried to achieve good cold-start recommendation performance.

Figure 23.7 shows the impact of the number of user and service groups. An optimal MAE performance is reached when the number of groups is 20 for both new users and new services. As the number of groups further increases, many smaller groups will be generated. Restricted by the group size, similar users or services may be spitted into different groups, which will lower the prediction accuracy.

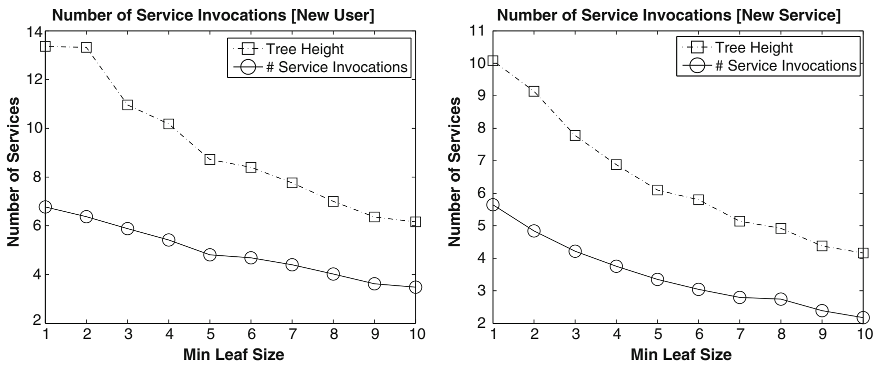


Fig. 23.6 Number of service invocations

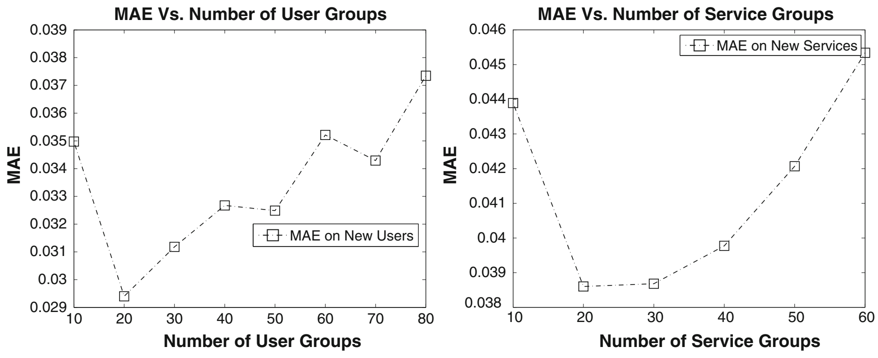


Fig. 23.7 Impact of number of user and service groups

23.5 Conclusion and Future Work

We develop a novel framework for bootstrapping service recommendation. The proposed framework offers a complete solution that tackles both new users and services. The framework is underpinned by Non-negative Matrix Tri-Factorization (NMTF) that simultaneously clusters users and services into a set of user and service groups. The group structure helps estimate the missing interaction information and also provides class labels to construct decision trees for both users and services. An initial interview is conducted to adaptively query users or services for rapid profiling. We propose to exploit surrogate users obtained from the user groups to improve the user response rate for profiling new services. The effectiveness of the proposed framework has been demonstrated via experiments on a real-world QoS dataset and through comparison with competitive collaborative filtering algorithms. An interesting future direction is to exploit existing work on reputation and trust management [7, 13] in service computing to get high-quality QoS data from users to further improve the quality of the recommendation result.

References

1. Chen, X., Liu, X., Huang, Z., Sun, H.: Regionknn: A scalable hybrid collaborative filtering algorithm for personalized web service recommendation. In: ICWS, pp. 9–16 (2010)
2. Chen, X., Zheng, Z., Liu, X., Huang, Z., Sun, H.: Personalized qos-aware web service recommendation and visualization. IEEE Trans. Serv. Comput. 99(PrePrints) (2011). http://doi.ieeecomputersociety.org/10.1109/TSC.2011.35
3. Golbandi, N., Koren, Y., Lempel, R.: On bootstrapping recommender systems. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, CIKM '10, pp. 1805–1808. ACM, New York (2010). doi:doi.acm.org/10.1145/1871437.1871734
4. Golbandi, N., Koren, Y., Lempel, R.: Adaptive bootstrapping of recommender systems using decision trees. In: Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, WSDM '11, pp. 595–604. ACM, New York (2011). doi:doi.acm.org/10.1145/1935826.1935910
5. Jiang, Y., Liu, J., Tang, M., Liu, X.F.: An effective web service recommendation method based on personalized collaborative filtering. In: ICWS, pp. 211–218 (2011)
6. Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. In: NIPS, pp. 556–562 (2000). http://citeseer.ist.psu.edu/lee01algorithms
7. Malik, Z., Bouguettaya, A.: Rateweb: Reputation assessment for trust establishment among web services. VLDB J. **18**(4), 885–911 (2009)
8. Rashid, A.M., Albert, I., Cosley, D., Lam, S.K., McNee, S.M., Konstan, J.A., Riedl, J.: Getting to know you: learning new user preferences in recommender systems. In: Proceedings of the 7th International Conference on Intelligent User Interfaces, IUI '02, pp. 127–134. ACM, New York (2002). doi:doi.acm.org/10.1145/502716.502737
9. Rashid, A.M., Karypis, G., Riedl, J.: Learning preferences of new users in recommender systems: an information theoretic approach. SIGKDD Explor. Newsl. **10**, 90–100 (2008). doi:doi.acm.org/10.1145/1540276.1540302
10. Rong, W., Liu, K., Liang, L.: Personalized web service ranking via user group combining association rule. IEEE Int. Conf. Web Serv. **0**, 445–452 (2009). doi:doi.ieeecomputersociety.org/10.1109/ICWS.2009.113

11. Shao, L., Zhang, J., Wei, Y., Zhao, J., Xie, B., Mei, H.: Personalized qos prediction forweb services via collaborative filtering. In: ICWS, pp. 439–446 (2007)
12. Wang, F., Li, T., Zhang, C.: Semi-supervised clustering via matrix factorization. In: SDM, pp. 1–12 (2008)
13. Yahyaoui, H., Zhioua, S.: Bootstrapping trust of web services through behavior observation. In: Auer, S., Díaz, O., Papadopoulos G.A. (eds.) ICWE, Lecture Notes in Computer Science, vol. 6757, pp. 319–330. Springer (2011)
14. Yu, Q.: Decision tree learning from incomplete qos to bootstrap service recommendation. In: ICWS '12: Proceedings of the 2012 IEEE International Conference on Web Services (2012)
15. Yu, Q.: Qos-aware service selection via collaborative qos evaluation (accepted to appear). The World Wide Web Journal (WWWJ) (2012) http://link.springer.com/article/10.1007%2Fs11280-012-0186-0
16. Yu, Q., Bouguettaya, A.: Framework for web service query algebra and optimization. TWEB **2**(1), 1–35 (2008)
17. Yu, Q., Rege, M., Bouguettaya, A., Medjahed, B., Ouzzani, M.: A two-phase framework for quality-awareweb service selection. Serv. Oriented Comput. Appl. **4**(2), 63–79 (2010)
18. Zhang, Q., Ding, C., Chi, C.H.: Collaborative filtering based service ranking using invocation histories. In: ICWS, pp. 195–202 (2011)
19. Zheng, Z., Ma, H., Lyu, M.R., King, I.: Wsrec: A collaborative filtering based web service recommender system. In: ICWS, pp. 437–444 (2009)
20. Zheng, Z., Ma, H., Lyu, M.R., King, I.: Collaborative web service qos prediction via neighborhood integrated matrix factorization. IEEE Trans. Serv. Comput. 99(PrePrints) (2011). doi:doi.ieeecomputersociety.org/10.1109/TSC.2011.35
21. Zhou, K., Yang, S.H., Zha, H.: Functional matrix factorizations for cold-start recommendation. In: Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '11, pp. 315–324. ACM, New York (2011). doi:doi.acm.org/10.1145/2009916.2009961