World Wide Web (2020) 23:241–265
https://doi.org/10.1007/s11280-019-00683-z

A meta-feature based unified framework for both cold-start and warm-start explainable recommendations

Ning Yang[1] · Yuchi Ma[1] · Li Chen[1] · Philip S. Yu[2,3]

Received: 29 June 2017 / Revised: 22 February 2019 / Accepted: 22 April 2019 /
Published online: 9 May 2019

Abstract

Recently, recommender systems have received an increasing amount of attention from researchers due to their indispensable role in the more and more popular e-commercial websites. Although a lot of methods have been proposed for warm-start recommendation, cold-start recommendation still remains open as one of the major challenges of recommender systems. The existing approaches often suffer from two defects. The first is the lack of *unified framework*. The existing researches often deal with the cold-start recommendation and the warm-start recommendation separately, which makes their respective methods hard to integrate into a system and keeps the cold-start users/items away from the existing ones. The second is the poor *interpretability*. The existing methods often ignore the complicated preferential relationships between users and item features, and can not quantitatively explain the multiple reasons that cause a user chooses an item. In this paper, we aim at the problem of making explainable recommendations for both warm-start and cold-start users/items in a unified framework, of which the challenges are three-fold, the lack of meaningful information, large-scale data, and quantitative explanation. To address these challenges, we propose a novel concept referred to as *meta-feature*, and a Meta-feature based Explainable Recommendation Framework (MERF). Meta-features are latent features about item features, which can reveal the preferential relationship between users and item features, not just the items.

✉ Li Chen
cl@scu.edu.cn

Ning Yang
yangning@scu.edu.cn

Yuchi Ma
scu.Richard.Ma@gmail.com

Philip S. Yu
psyu@uic.edu

[1] School of Computer Science, Sichuan University, Chengdu, China

[2] Department of Computer Science, University of Illinois at Chicago, Chicago, IL, USA

[3] Institute for Data Science, Tsinghua University, Beijing, China

MERF is able to make recommendations for both cold-start and warm-start users/items in a unified framework based on meta-feature. Especially, thanks to meta-feature, MERF can make cold-start recommendations requiring no historical rating records but just the item features. To make a recommendation with a quantitative explanation, we propose a Personalized Feature Preference (PFP) vector to characterize the different importance of item features to a user. MERF makes a recommendation based on an Item Rating Matrix and an Explanation Matrix, which can be estimated by fusing PFP and meta-features. To improve the efficiency of MERF, we also propose a parallel learning algorithm and an incremental updating algorithm for PFP. At last, extensive experiments conducted on real datasets verify the effectiveness and efficiency of the proposed approach.

Keywords Meta-feature · Cold-start problem · Explainable recommendation

1 Introduction

Recently, recommender systems have received an increasing amount of attention from researchers due to their indispensable role in the more and more popular e-commercial websites. Although a lot of methods have been proposed for warm-start recommendation, cold-start recommendation still remains open as one of the major challenges of recommender systems [8, 29, 41, 46]. The existing approaches often suffer from two defects. The first is the lack of *unified framework*. The existing researches often deal with the cold-start recommendation and warm-start recommendation separately, which makes their respective methods hard to integrate into a system and keeps the cold-start users/items away from the existing ones. The second is the poor *interpretability*. To make an explainable recommendation requires the methods not only to predict the ratings of users to items but also to generate an quantitative explanations for the preferences of users. The models used by the existing methods to predict the rating that a user would give to an item often ignore the complicated preferential relationships between users and item features, and can not quantitatively explain the multiple reasons that cause a user chooses an item.

In this paper, we aim at the problem of making cold-start and warm-start recommendations in a unified framework, which are quantitatively explainable. For example, for new items or existing items, we want to make a recommendation like *"We recommend the movie 'The Revenant' (4) to you, as we know you care the rating and the leading performer of a movie the most, its rating in IMDB is really high (1.5), and more importantly, its leading performer is your favorite (2.5)"*. Here the estimated rating of the user to the movie 'The Revenant' is 4 out of 5 stars, while the degrees to which the movie rating and leading performer of the movie satisfy the user preference are 1.5 and 2.5, respectively. In this paper, we want to build the explanation vector $\langle 1.5, 2.5 \rangle$, which quantifies the reasons why the rating is estimated as 4. Similarly, for new users or existing users, we want to make a recommendation like *"We recommend the movie 'The Hunger Games' (3) to you, as its rating in IMDB is really high (1.5), and its genre is your favorite (1.5)"*. However, making quantitatively explainable recommendations for both cold-start users/items and warm-start ones in a unified framework is not easy due to the following challenges.

- **Lack of meaningful information**: For a new user, her/his historical ratings often are insufficient to profile her/him, while for a new item, we may even have no historical ratings at all.
- **Large-scale data**: The existing methods like Collaborative Filtering [39] are often based on factorization of a matrix consisting of all the users and items. In real-world,

however, the numbers of users and items are usually very huge, which makes the existing methods too time-consuming to make a recommendation timely.

- **Quantitative explanation**: In real-world, the reasons motivating a user to choose an item are likely affected by multiple factors due to the complexity of the preference of an individual [10, 26]. We need to quantify and fuse the factors for making a recommendation with convincing explanation.

In this paper, we propose a Meta-feature based Explainable Recommendation Framework (MERF) to address the problem of making explainable cold-start and warm-start recommendations in a unified framework. The main idea of MERF is *to estimate the ratings of a user on new items by transferring the user's preferences to the features shared by the existing items with new items*. Based on this idea, we propose a novel concept, referred to as *meta-feature*. Roughly speaking, meta-features are features about item features. For example, the movie "The Phantom of the Opera" is "Exquisite", which is a description of the movie's genre feature of "Romance", and also a description of user character. Then we say the preferential relationship that "Exquisite" users often like "Exquisite" movies is a meta-feature of the movie genre feature "Romance". Meta-feature reveals the preferential relationship between users and item features (not items), which is different from traditional collaborative filtering based models that focus on the relationship between users and items. The promising advantage of meta-feature is that meta-feature makes MERF be able to generate the latent features of items without collaboration with historical ratings of users (which are unavailable for cold-start recommendations), which is in sharp contrast with the traditional methods where the item latent features depend on a collaborative factorization of a user historical rating matrix. Due to this advantage, making cold-start recommendations based on meta-feature requires no his/her historical rating records but just the item features.

People often pay attention of different degrees to different features of items. For example, when choosing a movie, some users may regard as important the genre of movies, while some other users may care more about the leading performer of movies. Inspired by this observation, we propose a Personalized Feature Preference (PFP) vector to characterize the different importance of item features to a user, and can be learned from the historical data of existing users and the profiles of new users. By fusing the PFP, item latent features, and user latent features, MERF estimates the Item Rating Matrix for all users over all items, and generates an Explanation Matrix for each user. At last, MERF makes a recommendation based on the estimated ratings of a user, and the corresponding row vector of the Explanation Matrix quantitatively explains why an item is recommended to that user.

To improve the efficiency of MERF, we propose a parallel algorithm for the learning of PFP, which can generate a PFP for each user concurrently. At the same time, as new ratings continuously become available, we also propose an incremental algorithm for the updating of PFP.

Our contributions can be summarized as follows:

(1) We propose a Meta-feature based Explainable Recommendation Framework (MERF) for the problem of making explainable cold-start and warm-start recommendations in a unified framework.
(2) We propose a novel concept of meta-feature to reveal the preferential relationship between users and item features, due to which MERF can make recommendations for cold-start users/items and warm-start ones in a consistent fashion.
(3) We propose a Personalized Feature Preference (PFP) vector to characterize the different importance of item features to a user. By fusing PFP and meta-features, MERF can make a recommendation with a quantitative explanation.

(4) To improve the efficiency, we propose a parallel learning algorithm and an incremental updating algorithm for PFP.
(5) We compare MERF with seven baseline methods on three real world datasets. The results verify the effectiveness and efficiency of MERF. We also verify the statistical significance of the superiority of MERF.

The rest of this paper is organized as follows. Section 2 gives an overview of the MERF. Section 3 describes the details of meta-feature. Section 4 describes the details of PFP and its learning and updating algorithms. Section 5 describes the details of MERF. Section 6 presents the experimental results and analysis. Finally, we discuss related works in Section 7 and conclude in Section 8.

2 Overview

Figure 1 gives an overview flow of MERF, where computation procedures are represented by black rectangles and data structures are represented by white rectangles. As shown in Figure 1, MERF can be divided into three stages which are described as follows.

- **Preparation**: The goal of preparation stage is to off-line generate latent feature matrices of users and meta-feature matrices of item features. MERF first fuses the historical ratings of existing users and the profiles of new users into Item Feature Rating Matrices (IFVRMs) of different item features, and an entry in IFVRM stores the rating of a user gives to a feature value of an item feature. MERF then generates a User Latent Feature Matrix and meta-feature matrices of different features by factorizing the IFVRMs.
- **Learning**: The goal of learning stage is to on-line generate PFPs, by minimizing the differences between the ground-truth ratings and the estimated ratings.
- **Recommendation**: The goal of this stage is to estimate the Item Rating Matrix for both existing and new users and both existing and new items, and generate an Explanation Matrix for each user. At first, based on the meta-feature matrices obtained from the first stage and the features of items (including new items), MERF generates an Item

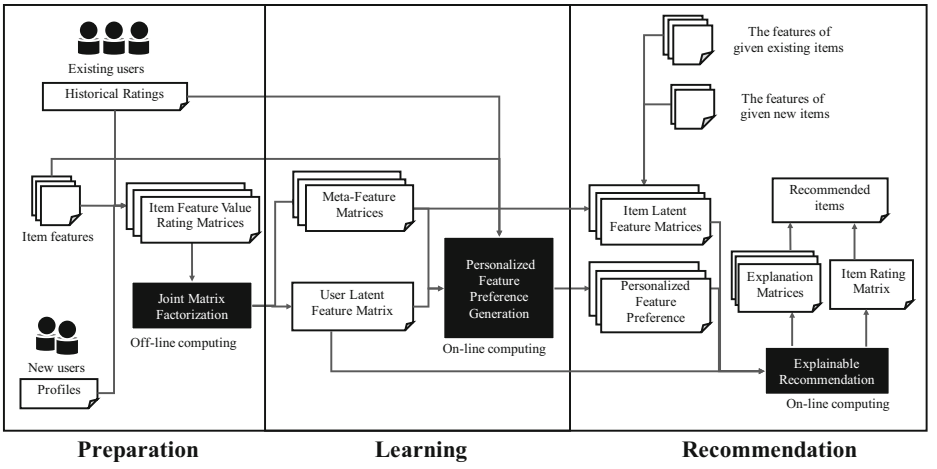


Figure 1 Overview of MERF

Latent Feature Matrix (ILFM) for each item. Then by fusing the PFP obtained from the second stage, the ILFM, and the User Latent Feature Matrix obtained from the first stage, MERF estimates the Item Rating Matrix for all users over all items, and generates an Explanation Matrix for each user. Here the key that MERF can serve both cold-start recommendations and warm-start ones lies in that the estimated item rating matrix contains ratings of both existing and new users to both existing and new items.

As new ratings are arriving continuously, the PFPs have to be updated in time. Thanks to meta-feature, a PFP can be updated with a constant time complexity for each new rating. In this way, MERF can make timely recommendations for the users whose preferences change frequently.

In this paper, a scalar is denoted by an italic capital letter (e.g., N), and a vector is denoted by a boldface lowercase letter (e.g., $\boldsymbol{v}$). A matrix is denoted by a boldface italic capital letter, e.g., $\boldsymbol{R} \in \mathbb{R}^{N \times M}$, and an entry in $\boldsymbol{R}$ is denoted by $\boldsymbol{R}_{i,j}$. A set is denoted by a italic capital letter (e.g., S). The main notations used throughout this paper are summarized in Table 1.

3 Meta-feature

In this section, we first give the definition of meta-feature, and then describe the details of how to learn meta-features, which corresponds to the first stage of MERF as shown in Figure 1.

3.1 Definition of meta-feature

As we have mentioned, a meta-feature is the latent feature of a feature of items, which is formally defined as follow.

Definition 1 (**Meta-feature**) The meta-feature matrix of the i-th feature of items is defined as a matrix $\boldsymbol{Q}^{(i)} \in \mathbb{R}^{\phi_i \times D}$, where ϕ_i is the number of different possible values of the i-th feature, and D is dimensionality of the meta-feature. The j-th row vector $\boldsymbol{Q}^{(i)}_{j,*}$ represents the meta-feature of the j-th possible value of the i-th feature.

Table 1 Summary of notations

N	Number of users
M	Number of existing items
L	Number of features of items
D	Dimensionality of meta-feature and user latent feature
ϕ_i	Number of different possible values taken by feature i
$\boldsymbol{R} \in \mathbb{R}^{N \times M}$	Item Rating Matrix
$\boldsymbol{F}^{(i)} \in \mathbb{R}^{M \times \phi_i}$	Item Feature Value Matrix of item feature i
$\boldsymbol{P}^{(i)} \in \mathbb{R}^{N \times \phi_i}$	Item Feature Value Rating Matrix of item feature i
$\boldsymbol{Q}^{(i)} \in \mathbb{R}^{\phi_i \times D}$	Meta-Feature Matrix of item feature i
$\boldsymbol{C}^{(i)} \in \mathbb{R}^{M \times D}$	Item Latent Feature Matrix of item feature i
$\boldsymbol{U} \in \mathbb{R}^{N \times D}$	User Latent Feature Matrix
$\boldsymbol{E}^{(u)} \in \mathbb{R}^{M \times L}$	Explanation Matrix of user u

One can note that the meta-feature is well defined over discrete features. However, it is easy to extend it to continuous features by quantizing their values into discrete levels.

3.2 Item feature value rating matrix

For generating a meta-feature matrix, we first introduce Item Feature Value Rating Matrix (IFVRM), which models the preference of users to a value of an item feature (not items themselves).

Definition 2 (**Item Feature Value Rating Matrix (IFVRM)**) The IFVRM of i-th feature is defined as a matrix $\boldsymbol{P}^{(i)} \in \mathbb{R}^{N \times \phi_i}$, where N is the number of users (including both existing users and new users), and an element $\boldsymbol{P}^{(i)}_{j,k}$ represents the quantity of the preference of the j-th user to the k-th possible value of the i-th feature of items.

According to Definition 2, an IFVRM $\boldsymbol{P}^{(i)}$ consists of two parts, i.e.,

$$\boldsymbol{P}^{(i)} = [(\boldsymbol{P}^{(i)}_{\mathrm{e}})^{\mathrm{T}}, (\boldsymbol{P}^{(i)}_{\mathrm{n}})^{\mathrm{T}}]^{\mathrm{T}}, \tag{1}$$

where $\boldsymbol{P}^{(i)}_{\mathrm{e}} \in \mathbb{R}^{N_{\mathrm{e}} \times \phi_i}$ and $\boldsymbol{P}^{(i)}_{\mathrm{n}} \in \mathbb{R}^{N_{\mathrm{n}} \times \phi_i}$ are the IFVRMs of existing users and new users, respectively, and N_{e} is the number of existing users while N_{n} is the number of new users.

The IFVRM of existing users, $\boldsymbol{P}^{(i)}_{\mathrm{e}}$, can be built as a product of the historical rating matrix $\boldsymbol{R} \in \mathbb{R}^{N_{\mathrm{e}} \times M}$ and the value matrix $\boldsymbol{F}^{(i)}$ of the i-th feature, i.e., $\boldsymbol{P}^{(i)}_{\mathrm{e}} = \boldsymbol{R}\boldsymbol{F}^{(i)}$. $\boldsymbol{F}^{(i)} \in \mathbb{R}^{M \times \phi_i}$, where an entry $\boldsymbol{F}^{(i)}_{j,v} = 1$ if the j-th item takes value v on the i-th feature, otherwise $\boldsymbol{F}^{(i)}_{j,v} = 0$. Figure 2 gives an illustration of feature value matrix, where "Director", "Starring", "Movie Genre" are movie features. The feature "Movie Genre" has 28 possible values which represent different genres of movies, for example, "Romance" and "Disaster". In the feature value matrix $\boldsymbol{F}^{(3)}$ of the feature "Movie Genre", the row for the movie "Titanic" takes the same value 1 on the genres "Romance" and "Disaster", as "Titanic" is a movie of "Romance" as well as "Disaster".

Initializing the preferences of cold-start users is still an open problem. Our idea in this paper is to alleviate it with the help of the preference patterns hidden in the profiles of users. We treat a preference, say "Liking Travel", as an item, and then the preferences recorded in a user's profile can be viewed as an item set. To build the IFVRM of new users, $\boldsymbol{P}^{(i)}_{\mathrm{n}}$, we apply association rule mining to the profiles of all users and the association rules can be considered as the feature preference of new users, where each rule shares the same weight.

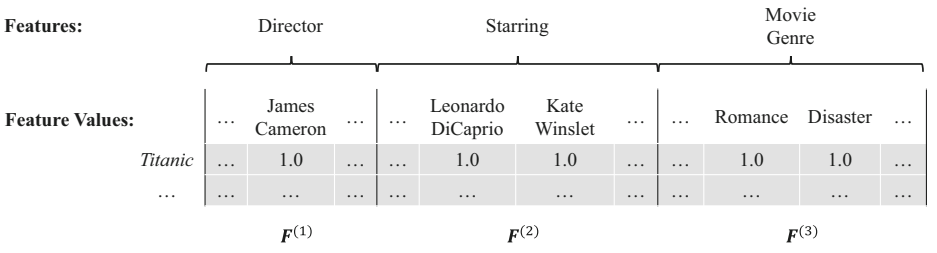


Figure 2 An illustration of feature value matrix

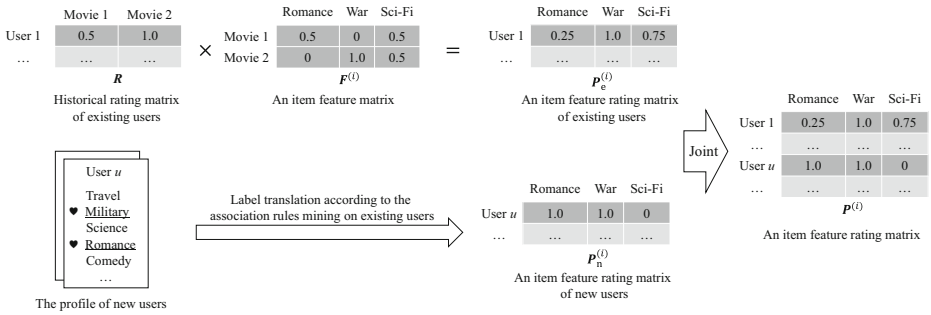


Figure 3 An illustration of how an IFVRM is built

For example, if one association rule is "Liking Travel → Liking Adventure Movie", and we know one new user likes travel from her/his profile, then we can infer that she/he also likes adventure movies, and the corresponding cell in $\boldsymbol{P}_{\mathrm{n}}^{(i)}$ of that user will be set to 1. Figure 3 gives an illustration of the building of IFVRM.

3.3 Learning meta-feature matrix

As we have mentioned, the meta-feature matrix $\boldsymbol{Q}^{(i)}$ is essentially the latent feature values of the i-th feature of items, and therefore the more similar the user latent feature is to the $\boldsymbol{Q}^{(i)}$, the greater values which the cells of $\boldsymbol{P}^{(i)}$ may take, i.e., the value of a cell $\boldsymbol{P}_{j,k}^{(i)}$ is proportional to the inner product of $\boldsymbol{U}_{j,*}$ and $\boldsymbol{Q}_{*,k}^{(i)}$, where $\boldsymbol{U} \in \mathbb{R}^{N\times D}$ is the user latent feature matrix. Based on this insight, the IFVRM $\boldsymbol{P}^{(i)}$ can be considered as the product of a user latent feature matrix $\boldsymbol{U} \in \mathbb{R}^{N\times D}$ and $\boldsymbol{Q}^{(i)}$, i.e., $\boldsymbol{P}^{(i)} \approx \boldsymbol{U}\,\boldsymbol{Q}^{(i)^{\mathrm{T}}}$. Then the meta-feature matrices $\boldsymbol{Q}^{(1)}, \boldsymbol{Q}^{(2)}, \cdots, \boldsymbol{Q}^{(L)}$ and the user latent feature matrix $\boldsymbol{U}$ can be learned by using gradient descent algorithm to solve the following optimization problem:

$$\underset{\boldsymbol{U},\boldsymbol{Q}^{(i)}}{\operatorname{argmin}} \frac{1}{2}\sum_{i=1}^{L} \|\boldsymbol{P}^{(i)} - \boldsymbol{U}\,\boldsymbol{Q}^{(i)^{\mathrm{T}}}\|_{\mathrm{F}}^2 + \frac{\lambda}{2}(\|\boldsymbol{U}\|_{\mathrm{F}}^2 + \sum_{i=1}^{L} \|\boldsymbol{Q}^{(i)}\|_{\mathrm{F}}^2), \tag{2}$$

where $\|*\|_{\mathrm{F}}$ denotes the Frobenius norm, L is the number of different item features, and λ is the balance parameter. In (2), the meta-feature matrix $\boldsymbol{Q}^{(i)}$ and the User Latent Feature Matrix $\boldsymbol{U}$ are to be learned, while the IFVRM $\boldsymbol{P}^{(i)}$ is obtained in advance through (1).

4 Personalized feature preference

People often pay different attention to different features of an item. For example, some people may care much about the director of a movie, while the others may care much about the genre of a movie. In this section, we first introduce a vector, Personalized Feature Preference, which reflects how much a user care about different features, and then describe the details of how to learn it. This part corresponds to the second stage of MERF as shown in Figure 1.

4.1 Definition of personalized feature preference

Definition 3 (**Personalized Feature Preference (PFP)**) The Personalized Feature Preference of user u is defined as an L-dimensional vector

$$\boldsymbol{\alpha}_u = \{\boldsymbol{\alpha}_u^{(1)}, ..., \boldsymbol{\alpha}_u^{(i)}, ..., \boldsymbol{\alpha}_u^{(L)}\}^{\mathrm{T}},$$

where the i-th component $\boldsymbol{\alpha}_u^{(i)} \geq 0$ is the personal preference weight of user u to the i-the feature, and $\sum_{i=1}^{L} \boldsymbol{\alpha}_u^{(i)} = 1$.

4.2 Learning personalized feature preference

Now we describe the details of how to learn the PFP of a user u. In order to generate personalized feature preference, we have to estimate the Item Latent Feature Matrix (ILFM) $\boldsymbol{C}^{(i)} \in \mathbb{R}^{M \times D}$ on each item feature i. One can not confuse $\boldsymbol{C}^{(i)}$ with $\boldsymbol{Q}^{(i)}$, where $\boldsymbol{C}^{(i)}$ is the latent feature matrix of item with respect to i-th feature, while the meta-feature matrix $\boldsymbol{Q}^{(i)}$ is the latent feature matrix of i-th feature.

Thanks to the meta-feature matrix, we can generate a latent feature vector on a specific feature for a new item, as well as for an existing item, by multiplying the specific feature of the item with the meta-feature matrix, for which an illustration is given by Figure 4. Formally, on the i-th feature, we can generate the corresponding item latent feature matrix by the product between the item feature value matrix and meta-feature matrix of the i-th feature as follow:

$$\boldsymbol{C}^{(i)} = \boldsymbol{F}^{(i)} \boldsymbol{Q}^{(i)}, \tag{3}$$

where $\boldsymbol{C}^{(i)} \in \mathbb{R}^{M \times D}$ is the item latent feature matrix on the i-th feature, $\boldsymbol{F}^{(i)} \in \mathbb{R}^{M \times \phi_i}$ and $\boldsymbol{Q}^{(i)} \in \mathbb{R}^{\phi_i \times D}$ are the value matrix and the meta-feature matrix of the i-th feature, respectively.

Once the ILFM $\boldsymbol{C}^{(i)}$ is generated, we can further estimate the Item Rating Matrix $\boldsymbol{R} \in \mathbb{R}^{N \times M}$, where a cell $\boldsymbol{R}_{u,m}$ represents the rating score that the u-th user gives to the m-th item. Intuitively, $\boldsymbol{R}_{u,m}$ depends on two factors. The first factor is the similarity $r_{u,m}^{(i)}$ between the user latent feature vector $\boldsymbol{U}_{u,*}$ and the item latent feature vector $\boldsymbol{C}_{m,*}^{(i)}$, $1 \leq i \leq L$, which can be evaluated by the following equation:

$$r_{u,m}^{(i)} = \boldsymbol{U}_{u,*}^{T} \boldsymbol{C}_{m,*}^{(i)}. \tag{4}$$

where $\boldsymbol{U}$ is the user latent feature matrix obtained through (2), and $\boldsymbol{C}^{(i)}$ is the item latent feature matrix of the i-th feature obtained through (3). The second factor is the importance

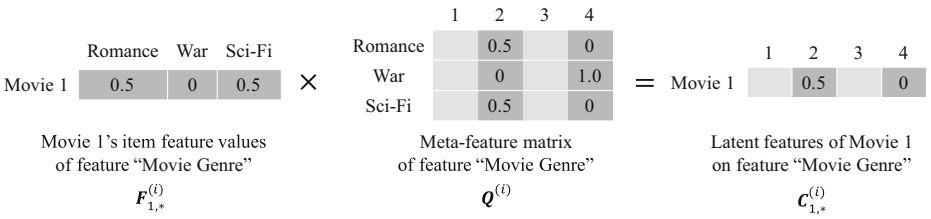


Figure 4 An illustration of generating item latent features from meta-features

of an item feature to the user, i.e., $\boldsymbol{\alpha}_u^{(i)}$. Based on this intuition, we can estimate the rating score of the u-th user to the m-th item as a weighted sum, i.e.,

$$\widehat{\boldsymbol{R}}_{u,m} = \sum_{i}^{L} \boldsymbol{\alpha}_u^{(i)} r_{u,m}^{(i)}. \tag{5}$$

Note that in (5), the $\boldsymbol{\alpha}_u^{(i)}$, $1 \leq i \leq L$, are the personalized feature preferences that are unknown and need to be learned.

Finally, we can learn the PFP $\boldsymbol{\alpha}_u$ by minimizing the difference between the ground-truth ratings $\boldsymbol{R}_{u,m}$ obtained from the ratings of users and estimated ratings $\widehat{\boldsymbol{R}}_{u,m}$ as follow:

$$\begin{aligned} &\operatorname{argmin}_{\boldsymbol{\alpha}_u^{(i)}} \ \tfrac{1}{2}\sum_{m}^{M}(\boldsymbol{R}_{u,m} - \widehat{\boldsymbol{R}}_{u,m})^2, \\ &s.t. \qquad \sum_{i=1}^{L} \boldsymbol{\alpha}_u^{(i)} = 1. \end{aligned} \tag{6}$$

By Lagrangian multiplier method, after few simple derivations, we obtain

Algorithm 1 Parallel personalized feature preference learning algorithm.

INPUT:

$\boldsymbol{R} \in \mathbb{R}^{N \times M}$: Item rating matrix;
$\boldsymbol{U}$: User latent feature matrix;
$\{\boldsymbol{Q}^{(1)}, \cdots, \boldsymbol{Q}^{(L)}\}$: Meta-feature matrices;
$\{\boldsymbol{F}^{(1)}, \cdots, \boldsymbol{F}^{(L)}\}$: Item feature value matrices;

OUTPUT:

$\{\boldsymbol{\alpha}_u\}$: PFP vectors;

```
for i = 1 to L {
    C^(i) = F^(i) Q^(i) according to (3);
}
    parallel foreach (user u) {
        for (i = 1 to L) {
            for (m = 1 to M) {
                Calculate α_u^(i) according to (7);
                Add α_u^(i) into α_u;
            }
        }
}
```

$$\boldsymbol{\alpha}_u^{(i)} = \frac{b_u^{(i)}}{c_u^{(i)}} + \frac{1}{L}, \tag{7}$$

where

$$b_u^{(i)} = L \sum_{m=1}^{M} \boldsymbol{R}_{u,m} r_{u,m}^{(i)} - \sum_{i=1}^{L}\sum_{m=1}^{M} \boldsymbol{R}_{u,m} r_{u,m}^{(i)}, \tag{8}$$

$$c_u^{(i)} = L \sum_{m=1}^{M}\sum_{i=1}^{L} (r_{u,m}^{(i)})^2. \tag{9}$$

Note that for any new user u, $\boldsymbol{R}_{u,m} = 0$, then $\boldsymbol{\alpha}_u^{(i)} = \frac{1}{L}$, which means for a new user u who does not have historical ratings to any item, the L item features share the same preference of the new user.

Algorithm 1 gives the parallel preference generating algorithm. The time complexity of Line (1) to Line (3) is $O(LD)$, where D is the number of latent features, L is the number of item features, which are all constants. Let H be the number of threads, the time complexity of Line (4) to Line (12) is $O(LDNM/H)$. Therefore, the overall time complexity of Algorithm 1 is $O(NM/H)$.

4.3 Updating personalized feature preference

Suppose a new rating $\boldsymbol{R}_{u,m}$ is observed, where u can be a new user or an existing user and m can be a new item or an existing item. Now we need to update the personalized feature preference $\boldsymbol{\alpha}_u$. At first, according to (8) and (9), we can incrementally update the numerator and denominator of the first term of (7) respectively as follows:

$$b_u'^{(i)} = b_u^{(i)} + L\boldsymbol{R}_{u,m} r_{u,m}^{(i)} - \sum_{i=1}^{L} \boldsymbol{R}_{u,m} r_{u,m}^{(i)}, \tag{10}$$

$$c_u'^{(i)} = c_u^{(i)} + L \sum_{i=1}^{L} (r_{u,m}^{(i)})^2. \tag{11}$$

Then we can incrementally update $\boldsymbol{\alpha}_u^{(i)}$ as the new one $\boldsymbol{\alpha}_u'^{(i)}$, i.e.,

$$\boldsymbol{\alpha}_u'^{(i)} = \frac{b_u'^{(i)}}{c_u'^{(i)}} + \frac{1}{L}. \tag{12}$$

Algorithm 2 Personalized feature preference updating algorithm.

INPUT: $\{\boldsymbol{\alpha}_u\}$: The old PFP vectors;
$\{\boldsymbol{R}_{u,m}\}$: The set of new ratings;
$\boldsymbol{U}$: User latent feature matrix;
$\{\boldsymbol{Q}^{(1)}, \cdots, \boldsymbol{Q}^{(L)}\}$: Meta-feature matrices;
$\{\boldsymbol{F}^{(1)}, \cdots, \boldsymbol{F}^{(L)}\}$: Item feature value matrices;

OUTPUT:
$\{\boldsymbol{\alpha}\prime_u\}$: The updated PFP vectors;

```
1: for i = 1 to L {
2:     C^(i) = F^(i) Q^(i) according to (3);
3: }
4:     foreach (new rating R'_{u,m}) {
5:         for (i = 1 to L) {
6:             Calculate α'^(i)_u according to (12);
7:             Add α'^(i)_u into α'_u;
8:     }
9: }
```

Algorithm 2 gives the updating algorithm. The time complexity of Lines from (2) to (4) is $O(LD)$, where D is the number of latent features and L is the number of item features, which are all constants. And the time complexity of Lines from (5) to (11) is

$O(LD|\{\boldsymbol{R}_{u,m}\}|)$. Thus, the time complexity of Algorithm 2 is linear in the number of new ratings, and for each new rating, Algorithm 2 can update the PFP with constant time complexity.

5 Explainable recommendation

In this section, we describe the details of how an explainable recommendation is made, which corresponds to the third stage of MERF as shown in Figure 1.

Algorithm 3 Parallel explainable recommendation algorithm.

INPUT:

u: User to whom we want to recommend;
$\boldsymbol{R}$: Item rating matrix;
$\boldsymbol{U}$: User latent feature matrix;
$\{\boldsymbol{C}^{(1)}, \cdots, \boldsymbol{C}^{(L)}\}$: Item latent feature matrices;
$\boldsymbol{\alpha}_u$: Personalized feature preference vector;

OUTPUT:

m_u: the recommended item;
$\boldsymbol{E}^{(u)}$: Explanation matrix;

```
1: parallel foreach (user u) {
2:     m_u = 1≤m≤M R_{u,m}, according to (13).
3:     for (m = 1 to M) {
4:         for (i = 1 to L) {
5:             E^(u)_{m,i} = α^(i)_u U^T_{u,*} C^(i)_{m,*}, according to (14);
6:     }
7:   }
8: }
```

5.1 Making recommendation

So far, based on meta-feature $\boldsymbol{Q}^{(i)}$, we have generated the user latent feature matrix $\boldsymbol{U}$ (via (2)), the item latent feature matrix $\boldsymbol{C}^{(i)}$ for both existing items and new items (via (3)), and the personalized feature preference $\boldsymbol{\alpha}_u$ for both existing users and new users (via (6)). Then we estimate the item rating matrix $\boldsymbol{R} \in \mathbb{R}^{N\times M}$ using (5), and for a user u, the item m_u which has highest rating score is recommended to u, i.e.,

$$m_u = \underset{1\le m\le M}{\arg\max}\ \boldsymbol{R}_{u,m}. \tag{13}$$

5.2 Explaining recommendation

As we have mentioned, on one hand, more similar the latent feature vector $\boldsymbol{U}_{u,*}$ of a user u is to the item latent feature vector $\boldsymbol{C}^{(i)}_{m,*}$ of feature i of item m, more possible the feature i of m partially causes the user u chooses the item m. On the other hand, different features have difference importance to a user, which is captured by the PFP vector $\boldsymbol{\alpha}_u$. Based on these

insights, the quantitative explanation of a user u chooses item m due to feature i, denoted by $\boldsymbol{E}_{m,i}^{(u)}$, can be modeled as:

$$\boldsymbol{E}_{m,i}^{(u)} = \boldsymbol{\alpha}_u^{(i)} r_{u,m}^{(i)} = \boldsymbol{\alpha}_u^{(i)} \boldsymbol{U}_{u,*}^T \boldsymbol{C}_{m,*}^{(i)}. \tag{14}$$

Essentially, $\boldsymbol{E}_{m,i}^{(u)}$ represents the degree to which the item feature i satisfies the user preference. Based on these quantitative explanations, for a user u we can build the explanation matrix, $\boldsymbol{E}^{(u)} \in \mathbb{R}^{M \times L}$, as follow:

$$\boldsymbol{E}^{(u)} = \begin{Bmatrix} \boldsymbol{E}_{1,1}^{(u)} & \dots \boldsymbol{E}_{1,i}^{(u)} \dots & \boldsymbol{E}_{1,L}^{(u)} \\ \dots & \dots & \dots \\ \boldsymbol{E}_{m,1}^{(u)} & \dots \boldsymbol{E}_{m,i}^{(u)} \dots & \boldsymbol{E}_{m,L}^{(u)} \\ \dots & \dots & \dots \\ \boldsymbol{E}_{M,1}^{(u)} & \dots \boldsymbol{E}_{M,i}^{(u)} \dots & \boldsymbol{E}_{M,L}^{(u)} \end{Bmatrix}. \tag{15}$$

The row vector $\boldsymbol{E}_{m,*}^{(u)}$ is exactly the explanation vector of why an item m is recommended to user u. As for the examples mentioned in Section 1, the explanation vector of the i-th movie 'The Relevance' is recommended to u would be $\boldsymbol{E}_{i,*}^{(u)} = \langle 1.5, 2.5 \rangle$, while the explanation vector of the j-th movie 'The Hunger Games' is recommended to user v would be $\boldsymbol{E}_{j,*}^{(v)} = \langle 1.5, 1.5 \rangle$.

Algorithm 3 gives the parallel explainable recommendation algorithm. The time complexity of Line (2) is $O(M)$, where M is the number of items. The time complexity of the loop (Lines from (3) to 6)) is $O(ML)$, where L is the number of item features. So the overall time complexity of Algorithm 3 is $O(M(1+L))$. In practice, the number of item features is small and can be treated as a constant, therefore the overall time complexity of Algorithm 3 is approximately $O(M)$.

6 Experiments

In this section, we compare our method MERF with baseline methods to verify its effectiveness and the statistical significance of its superiority, and verify the efficiency of MERF. All experiments are executed on a Windows 7 PC with an Intel Xeon CPU E3-1231 V3 (3.4GHz) and 16 GB RAM, and all programs are implemented in C#.

6.1 Datasets

We conduct our experiments on three real datasets, including Netlifx Dataset, IMDB dataset, and MovieTaste dataset collected from a movie recommendation website built by us. We align these datasets according to the movie titles, and conduct our experiments on the aligned dataset.

- **Netflix Dataset** [31] Netflix Dataset contains about more than 100 million rating records from about 480,000 users over about 17,000 movie titles. In this paper, we only consider users who have at least 10 rating records.
- **IMDB Dataset** [21] We crawled movie features from IMDB including rating, screened year, movie genre, director, leading performers and actresses. After aligning the movie titles between IMDB and the Netflix Dataset, we finally get 6 features of movies, namely movie rating, year, genre, director, leading performer and leading actress.

- **MovieTaste Dataset** [30] To collect the data of new users and new movies, we build a movie recommendation website, called MovieTaste. For each registered user, MovieTaste collects their label, and then records their feedbacks of randomly selected 100 movies from IMDB Top-250 movies. Besides, we select 167 new movies which are not in the Netflix Dataset and IMDB dataset, and add these movies into the candidate set of movie recommendation.The 167 new movies are considered as the cold-start items.

6.2 Metrics

We use two standard performance metrics for evaluating the performance of our approaches and baseline methods:

- **Normalized Discounted Cumulative Gain (NDCG)** NDCG is commonly used in information retrieval and recommender system [22]. A higher NDCG value for a list of recommended items indicates that more relevant items are ranked higher in the list. In particular, $NDCG@K$ measures the relevance of Top-K results and is defined as:

$$NDCG@K = \frac{DCG@K}{iDCG@K}, \tag{16}$$

  where $iDCG@K$ is the $DCG@K$ value of ideal ranking list; $DCG@K = rel_1 + \sum_{i=2}^{K} \frac{rel_i}{log_2(i+1)}$; $rel_i = 2^{r(i)} - 1$ is a relevance value, and $r(i)$ is the ground-truth ranking of the i-th item in the recommended list.
- **Hit Ratio (HR)** HR is widely used in top-K recommendation evaluation [17, 27, 48]. Recommendation can also be regarded as a supervised learning problem. For a user, we label the items as the positive examples if they are rated over median rating of the user, and negative examples, otherwise. Thus, the HR of a Top-K recommendation can be calculated as:

$$HR@K = \frac{1}{N} \sum_{u=1}^{N} \frac{|S_K^{(u)} \cap P^{(u))}|}{K}, \tag{17}$$

  where $P^{(u)}$ denotes the set of positive examples of user u, $S_K^{(u)}$ denotes the set of top-K rated items recommended to user u, and N is the number of users.
- **Agreement Ratio (AR)** We define AR to assess the explainability of MERF. Let $S_K^{(u)}$ be the set of top-K items recommended to user u, then AR is defined as:

$$AR@K = \frac{1}{N} \sum_{u=1}^{N} \left\{ \frac{1}{K} \sum_{m \in S_K^{(u)}} g(\boldsymbol{E}_{m,*}^{(u)}) \right\}, \tag{18}$$

  where $g(\boldsymbol{E}_{m,*}^{(u)}) = 1$ if feature l, $1 \le l \le L$, is one of the features the user u care about and $\boldsymbol{E}_{m,l}^{(u)} = \max\{\boldsymbol{E}_{m,1}^{(u)}, \cdots, \boldsymbol{E}_{m,L}^{(u)}\}$, otherwise $g(\boldsymbol{E}_{m,*}^{(u)}) = 0$. By $AR@K$, we can check whether the features of a recommended item satisfy the preferences of the user.

6.3 Baseline methods

We use 13 baseline methods in all, which are divided into three groups for the performance investigation of warm-start-recommendation, cold-start recommendation, and explainability, respectively. To set the hyper-parameters of baseline methods, on each dataset, we randomly select 60% of the data as training set, 20% as validating set for hyper-parameter

tuning, and the remaining 20% as testing set. We repeat this random partition five times and report the average of the results of the five runs as the final result.

6.3.1 Baseline methods for warm-start recommendation

In order to demonstrate the performance of MERF on warm-start recommendation, we will compare it with the following 7 baseline methods.

1. **Naive MERF** In order to validate the effectiveness of personalized feature preference, we use a Native MERF where all the feature preference parameters of users are set to 1.
2. **IBFM** IBFM is an Integrated Bias and Factorization Model proposed by Lu et al. [29], which can incorporate user and item's attributes together with the latent features of users and items.
3. **UBM** UBM is a Unified Bias Model which is a baseline method mentioned in [29]. UBM can make a recommendation by integrating all the aspects of items' features.
4. **UserCF** UserCF (User-based Collaborative Filtering) is a classical method for recommendation [39], which makes recommendations based on the similarity among users. To apply UserCF in the experiments, we compute the user similarity matrix based on the movie rating records.
5. **ItemCF** ItemCF (Item-based Collaborative Filtering) is another classical recommending method [39], which makes recommendations based on the similarity among items. To apply ItemCF in the experiments, we evaluate the item similarity matrix using the cosine function of the feature vectors of movies.
6. **MF** MF(Matrix Factorization) is proposed by Funk [14] to solve the movie recommendation problem in Netflix Price. MF assumes the latent features of objects are modeled by vectors, and different types of objects have factors with the same size.
7. **BiasedMF** BiasedMF (Biased Matrix Factorization) is proposed by Paterek [33], which is an extension of MF.
8. **FM** FM (Factorization Machine) [36] is a general predictor which models all interactions between variables using factorized parameters, including the interactions between latent factors of item features.
9. **SLIM** SLIM is a sparse linear model for Top-N recommendation [32], which learns a sparse coefficient matrix for the items in the system solely from the user purchase/rating profiles by solving a regularized optimization problem.

6.3.2 Baseline methods for cold-start recommendation

In order to demonstrate the performance of MERF on cold-start recommendation, we will compare it with the following 3 baseline methods which focus on cold-start problem.

1. **CSEL** CSEL is a context-aware semi-supervised co-training method for cold-start recommendation proposed by Zhang et al. [53], which uses a factorization model to capture fine-grained user-item context.
2. **DecRec** DecRec is a matrix completion based model proposed by Barjasteh et al. [3], which simultaneously exploits the similarity information among users and items to alleviate the cold-start problem.
3. **RAPARE** RAPARE is a rating comparison based model proposed by Xu et al. [49], which provides a fine-grained calibration on the latent profiles of cold-start users/items by exploring the differences between cold-start and existing users/items.

4. **LoCo** LoCo is a model for cold-start recommendation based on linear regression, low-rank parametrization, and randomized SVD [40].

6.3.3 Baseline methods for explainable recommendation

In order to demonstrate the explainability of MERF, we will compare it with the following 3 baseline methods which focus on explainable recommendation.

1. **EFM** EFM (Explicit Factor Model) is able to generate explainable recommendations based on explicit product features and user opinions extracted by phrase-level sentiment analysis on user reviews [54].
2. **TriRank** TriRank is a generic algorithm for ranking on tripartite graphs, which is specialized for personalized recommendations with explanations [17].
3. **sCVR** sCVR (Social Collaborative Viewpoint Regression) is a latent variable model which can predict user ratings by jointly modeling concepts, topics, sentiment labels and social relations as explanations [35].

6.4 Warm-start recommendation performance of MERF

We first examine the performance of MERF on warm-start recommendation for existing users, using the aligned dataset consisting of Netflix and IMDB. We split the movies into two groups, where one group consists of the top-250 movies (TOP-250) and the other the remaining movies (NON-TOP-250), and we observe the performances on the two groups, respectively. On each group, the ratings of each user are sorted chronologically, and the first 80% are used for training while the remaining 20% for testing. Table 2 shows the results on TOP-250 and Table 3 the results on NON-TOP-250, from which we can have the following observations:

(1) On TOP-250, MERF outperforms all the baseline methods. MERF performs better than MF, BiasedMF, SLIM, UserCF and ItemCF, which verifies the advantage that MERF can incorporate user and item's attributes together with the latent features of users and items. Next, MERF also outperforms IBFM and UBM, as MERF not only

Table 2 Performance of warm-start recommendation on TOP-250

	$HR@K$				$NDCG@K$			
	$K=4$	$K=6$	$K=8$	$K=10$	$K=4$	$K=6$	$K=8$	$K=10$
MERF	0.7963	0.8039	0.8128	0.8260	0.3744	0.3793	0.3881	0.4038
Naive MERF	0.7649	0.7758	0.7891	0.8021	0.3341	0.3411	0.3507	0.3641
IBFM	0.6967	0.7090	0.7204	0.7330	0.2874	0.2883	0.2895	0.2895
UBM	0.6917	0.7039	0.7147	0.7266	0.2813	0.2822	0.2838	0.2840
Basic MF	0.6858	0.6968	0.7062	0.7146	0.2556	0.2604	0.2638	0.2660
Biased MF	0.6808	0.6920	0.7016	0.7099	0.2597	0.2622	0.2646	0.2664
SLIM	0.7033	0.7108	0.7194	0.7338	0.3430	0.3489	0.3629	0.3852
UserCF	0.1986	0.3009	0.3588	0.4380	0.1667	0.2436	0.3169	0.5635
ItemCF	0.2975	0.3545	0.4317	0.5535	0.1601	0.1906	0.2337	0.3040
FM	0.1775	0.2579	0.3214	0.3878	0.2335	0.2568	0.2712	0.2801

Table 3 Performance of warm-start recommendation on NON-TOP-250

	$HR@K$				$NDCG@K$			
	$K=4$	$K=6$	$K=8$	$K=10$	$K=4$	$K=6$	$K=8$	$K=10$
MERF	0.7508	0.7889	0.8030	0.8271	0.3155	0.3163	0.3352	0.3588
Naive MERF	0.6911	0.7087	0.7266	0.7439	0.2911	0.3072	0.3117	0.3306
IBFM	0.6290	0.6301	0.6448	0.6577	0.2619	0.2729	0.2804	0.2879
UBM	0.6033	0.6179	0.6220	0.6396	0.2570	0.2582	0.2755	0.2796
Basic MF	0.5518	0.5596	0.5651	0.5802	0.1995	0.2018	0.2099	0.2238
Biased MF	0.5140	0.5293	0.5306	0.5380	0.2130	0.2281	0.2417	0.2590
SLIM	0.5079	0.5296	0.5384	0.5578	0.2305	0.2488	0.2502	0.2719
UserCF	0.1447	0.2118	0.2633	0.2850	0.1881	0.2029	0.2533	0.3885
ItemCF	0.2221	0.2298	0.2943	0.3655	0.1307	0.1953	0.2117	0.3101
FM	0.1255	0.1625	0.2495	0.2743	0.1943	0.2081	0.2400	0.2469

incorporates more attributes of users and items than IBFM and UBM, but also takes the difference between individuals into consideration. At last, we can see that although FM also considers the interactions of latent factors of items, the performance of FM is significantly far behind that of MERF, partly due to the defect that FM cannot learn the user different preferences to different item features. In contrast to FM, MERF is able to capture the different preferences of users to different item features by learning a Personalized Feature Preference (PFP) vector for each user.

(2) The performance of MERF and the baseline methods on TOP-250 is better than those on NON-TOP-250. We believe that this is because the rating records in NON-TOP-250 are sparser than those in TOP-250. We can also note, however, that even on NON-TOP-250, MERF still outperforms the baseline methods. This result not only shows again the advantages of MERF mentioned above, but also suggests that MERF is more robust for sparse historical rating data, as MERF is able to generate latent features of users and items based on not the historical ratings but meta-feature.

(3) MERF performs better than Naive MERF on both two groups, which verifies the contribution of PFP (Personalized Feature Preference) to the estimation of item ratings. As we have claimed, PFP captures the different importance of item features to users, and fusing PFP will lead to a more accurate estimation of ratings.

6.5 Cold-start recommendation performance of MERF

We then examine the performance of MERF on cold-start recommendation, using the MovieTaste dataset as new users and new items. The results are shown in Table 4, from which we can observe that MERF is also the best compared to the baseline methods. One can also note that performance of MERF on cold-start recommendation is better than that on warm-start recommendation. It is because users usually make high ratings to movies of high quality, and all the new movies are top-250 movies in IMDB. As a result, the low ratings on high quality movies reflect preference of users more significantly than any other cases, which explains the difference of performance between cold-start recommendation and warm-start recommendation.

Table 4 Performance of cold-start recommendation

	$HR@K$				$NDCG@K$			
	$K=4$	$K=6$	$K=8$	$K=10$	$K=4$	$K=6$	$K=8$	$K=10$
MERF	0.8103	0.8201	0.8367	0.8711	0.6243	0.6705	0.7045	0.7262
Naive MERF	0.7897	0.8018	0.8200	0.8477	0.5842	0.6463	0.6863	0.7021
PAPARE	0.7910	0.7938	0.8333	0.8475	0.6195	0.6482	0.6781	0.6837
CSEL	0.7870	0.7917	0.7949	0.8056	0.6205	0.6496	0.6737	0.6823
DecRec	0.7210	0.7238	0.7350	0.7950	0.5646	0.6042	0.6311	0.6480
LoCo	0.6909	0.6920	0.7011	0.7275	0.5004	0.5128	0.5242	0.5502

6.6 Explainability

To assess the explainability of MERF and the baseline methods, we first randomly choose $N = 100$ users who have profiles from which we can know the item features the users care about, and then make top-K recommendations to them. For each user, we mark item features as the features she/he cares about, if she/he chooses them as her/his preferences in her/his profile. For example, if a user chooses disaster movie as one of the preferences in the profile, we know the ground truth that this user likes disaster movie. The metric $AR@K$ (defined in (18)) can check whether the features of the item recommended by MERF satisfy the known preferences of the user by comparing the matrix $\boldsymbol{E}^{(u)}$ (defined in (15)) with the ground truths.

Table 5 shows the $AR@K$ of MERF and the baseline methods. At first, we can find that MERF outperforms all the baseline methods, which indicates that due to meta-features, PFP and Explanation Matrix, MERF captures the user attention to different features better than the baseline methods do. Besides, we can also observe the decline trend of $AR@K$ with increasing K. This is because as K increases, the value of the denominator of (18) grows faster than that of the numerator.

6.7 Test of significance

Next, we do the Friedman test [6] to show the superiority of our method. We run MERF and baseline methods 10 times for $K = 4, 6, 8$ and 10, respectively, where we randomly exchange 5% data of training set and test set at each time. Let r_j^i be the rank of the j-th method on the i-th test. The Friedman test compares the average ranks of our methods and the baseline methods, where the average rank of the j-th method is $R_j = \frac{1}{N}\sum_i r_j^i$, where $N = 40$ is the number test in our case. The null-hypothesis of Friedman test states all the

Table 5 Performance of explainability

	$AR@K$			
	$K=4$	$K=6$	$K=8$	$K=10$
MERF	0.713	0.697	0.671	0.633
sCVR	0.620	0.537	0.558	0.490
TriRank	0.451	0.487	0.377	0.352
EFM	0.473	0.417	0.384	0.301

algorithms are equivalent, and so their ranks R_j should be equal [9]. Iman and Davenport [38] show that the original Friedman statistic χ_F^2 is undesirably conservative and propose a better statistic

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2}, \tag{19}$$

where k is the number of methods; $\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right]$. In our case, $N = 10 \times 4$ and $k = 8$, thus the rejection region of null hypothesis at 95% confidence level is $F_F > \mu_{0.95} = 2.04$. Based on (19) and the results shown in Table 6, we get $\chi_F^2 = 277.53$ and $F_F = 4388.03 > 2.04$ for HR of existing users, $\chi_F^2 = 279.37$ and $F_F = 17203.11 > 2.04$ for NDCG of existing users, $\chi_F^2 = 217.47$ and $F_F = 135.63 > 2.04$ for HR of cold-start users, and $\chi_F^2 = 217.47$ and $F_F = 135.63 > 2.04$ for NDCG of cold-start users. Thus, the test statistics fall into the rejection region, which indicates that the alternative hypothesis, the average accuracies of our method and the baseline methods are not equivalent, is accepted.

Since the alternative hypothesis of Friedman test is accepted, we further proceed with a post-hoc test, Bonferroni-Dunn test [11], to make a further comparison between MERF and each baseline method. The performances of a pair of compared methods are significantly different if the corresponding average ranks differ by at least the critical difference

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}}, \tag{20}$$

where the critical value q_α is based on the Studentized range statistic divided by $\sqrt{2}$ [9]. By comparing the performance of MERF with those of the baseline methods, we calculate the average rank differences between MERF and baseline methods. As shown in Table 7, all the differences are greater than the critical difference $CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} = 1.66$, where $k = 8$, $N = 40$, and $q_\alpha = 3.03$, at 95% confidence level. As a result, MERF is superior to all the baseline methods at 95% confidence level.

In summary, MERF is an effective method, and the performance of MERF is better than all the baseline methods.

Table 6 The average rank comparison

	Warm-start recommendation		Cold-start recommendation	
	(HR)	(NDCG)	(HR)	(NDCG)
MERF	1.00	1.00	1.05	1.05
Naive MERF	2.00	2.00	2.00	2.00
IBFM	3.05	3.00	2.95	2.95
UBM	3.95	4.00	3.95	3.95
Basic MF	5.10	5.95	4.10	4.10
Biased MF	5.90	5.05	5.95	5.95
UserCF	7.05	7.00	8.00	8.00
ItemCF	7.95	8.00	7.00	7.00

Table 7 The rank differences between MERF and baseline methods

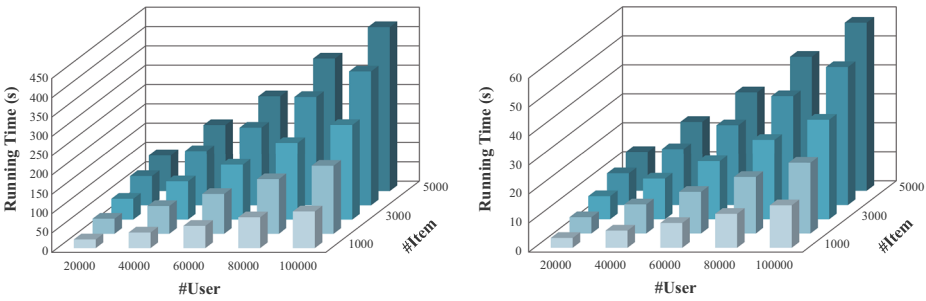
	Warm-start recommendation		Cold-start recommendation	
	(HR)	(NDCG)	(HR)	(NDCG)
IBFM	2.05	2.00	1.90	1.90
UBM	2.95	3.00	2.90	2.90
Basic MF	4.10	4.95	3.05	3.05
Biased MF	4.90	4.05	4.90	4.90
UserCF	6.05	6.00	7.00	7.00
ItemCF	6.95	7.00	5.95	5.95

6.8 Efficiency

To verify the efficiency of MERF, we first run our learning and recommendation algorithms with 8 threads. Figure 5 shows the running time of parallel personalized feature preference learning algorithm (Algorithm 1) and parallel explainable recommendation algorithm (Algorithm 3) are both linear in the number of users and items, which verifies the correctness of time complexity analysis of Algorithm 1 and Algorithm 3.

Next, we fix the number of users and items ($\#User = 20000, \#Item = 1000$), and investigate the efficiency of Algorithm 1 and Algorithm 3 with different number of threads. We use T threads to run the algorithms by setting $T = 2$, $T = 4$, $T = 6$, $T = 8$, respectively. Figure 6a indicates that the running time of the two algorithms decreases with the increase of the number of threads.

We also run the personalized feature preference updating algorithm (Algorithm 2) with different number of new ratings and a fixed number of users and items ($\#User = 20000, \#Item = 1000$), and record the running time. As shown in Figure 6b, the running time of Algorithm 2 is linear in the number of new ratings, which verifies the correctness of time complexity analysis of Algorithm 2.



(a) Parallel personalized feature preference learning

(b) Parallel recommendation

Figure 5 The running time of personalized feature preference generating and parallel recommendation algorithm

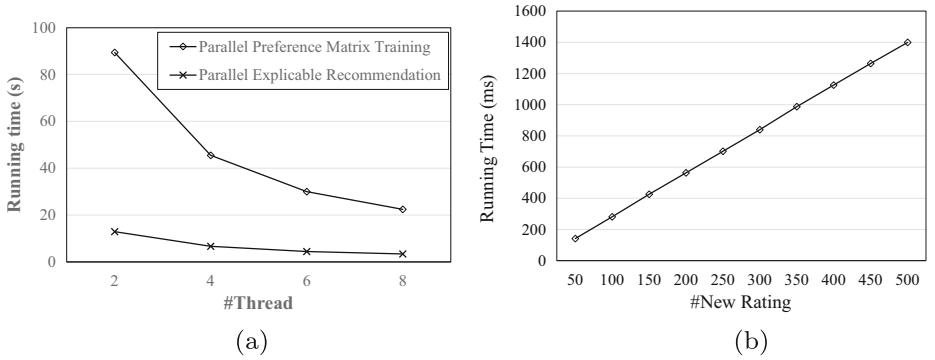


Figure 6 **a** The running time of parallel personalized feature preference learning (Algorithm 1) and parallel explainable recommendation (Algorithm 3) vs. different number of threads; (b)The running time of personalized feature preference updating (Algorithm 2) vs. the number of new ratings

7 Related works

In this section, we briefly introduce the related works about recommendation based on latent feature and explainable recommendation.

7.1 Latent feature based recommendation

Zheng et al. [56] present an approach, called User-Centered collaborative Location and Activity Filtering (UCLAF), which applies collaborative filtering to find like-minded users and like-patterned activities at different locations. UCLAF models the user location-activity relations as a three-order tensor and uses a context-aware tensor decomposition to address the sparse data problem in mobile information retrieval. Elbadrawy et al. [13] introduce a User-specific Feature-based Similarity Models (UFSM), where a linear similarity function is estimated for each user that depends entirely on features of the items previously liked by the user, which is then used to compute a score indicating how relevant a new item will be to that user. Hu et al. [19] propose a generalized Cross Domain Triadic Factorization (CDTF) model based on the triadic relation user-item-domain, where the users' rating on different domain's items are modeled by a three-order tensor denoting the dimensions of user, item and domain. In this way, CDTF can capture the interactions between domain-specific user factors and item factors. Li et al. [28] integrate a latent space matching procedure and a refining process to identify the matching, and propose a transfer-based method to improve the recommendation performance.

To overcome the issue of the sparsity of user-item rating data, Kim et al. [25] introduce a latent factor modeling method which exploits not only contextual information but also the numbers of ratings given to items. Yu et al. [52] propose a matrix factorization recommendation model which integrates social network information like trust relationships, rating information of users and users' own knowledge. Different from previous works, such recommendation model can make recommendations considering the user's own knowledge or expertise in a recommending field. To address the issue of the diversity of friend

recommendations in signed social networks, Agarwal et al. [1] introduce an adaptive consensus based framework which balances the diversity and consensus of the recommendations by employing a variable-length genetic algorithm. Jiang et al. [23] present a real-world job recommender system where a probabilistic model is proposed to cluster users based on the click behavior of users and make the job recommendations based on the learned corresponding prediction models for individual clusters.

A few of latent feature based methods have also been proposed for cold-start recommendation. Lu et al. [29] propose an Integrated Bias and Factorization Model (IBFM) for business recommendations, which employs a sampling strategy to generate the factor vectors for new users and new businesses based on similar users/businesses. Zhao et al. [55] present a Behavior Factorization model which separately models users' topical interests that come from various behavioral signals in order to construct better user profiles, and further employs matrix factorization techniques to model each user's behaviors as a separate example entry in the input user-by-topic matrix. Gantner et al. [15] propose a method that maps users or items to a latent feature space with a matrix factorization, by which the factors of a MF model trained by standard techniques can be applied to the new-user and the new-item problem. Gunawardana et al. [16] propose a Boltzmann machine based probabilistic model which encodes collaborative and content information as latent features so that information of different types is automatically combined. Barjasteh et al. [3] propose a general algorithmic framework based on matrix completion that simultaneously exploits the similarity information among users and items to alleviate the cold-start problem. Xu et al. [49] propose a rating comparison strategy (RAPARE) to learn the latent profiles of cold-start users/items, which provide a fine-grained calibration on the latent profiles of cold-start users/items by exploring the differences between cold-start and existing users/items. Silva et al. [44] make a hypothesis that users consumption preferences are biased to non-popular items. Based on this hypothesis, in [44], the authors propose two recommender algorithms to address the pure cold-start problem based on the user coverage maximization.

A few works that can learn latent features from information networks for cold-start recommendation have also been reported recently. Sedhain et al. [40] propose a learning based approach for the cold-start problem to leverage social network data via randomized SVD. Zhu et al. [58] propose a recommendation model which integrates the auxiliary information in multiple heterogeneous information networks (HINs) and transfers item latent information from different networks to help the recommendation task in a given network. Duricic et al. [12] propose an approach to generate a similarity matrix that is used to select the k-nearest neighbors for recommending items, by applying regular equivalence to a trust network where edges represent explicit or implicit trust relationships between users.

Recently, inspired by the power of deep learning, some works that apply neural networks to learn the latent nonlinear representations of users and items for recommendations have also been reported. Basically, these works fall into four categories, i.e., CNN based methods [20, 25, 51], auto-encoder based methods [24, 50], Recurrent Neural Network (RNN) based methods [4, 47], and attention network based methods [7, 42, 45]. Specifically, in [42], the authors propose an attention-based model named Attentional Content & Collaborate Model (ACCM), which adaptively integrates the Content-Based (CB) recommendation and Collaborative Filtering (CF) based recommendation. To make ACCM also work for cold-start scenarios, ACCM is learned via a cold sampling strategy which can simulate the cold-start users or items by randomly shadowing the historical feedback information of some users and items.

7.2 Explainable recommendation

Recent years, many researches aim at providing explanations in the context of recommendation systems. Most existing methods for explainable recommendation apply topic models to analyze user reviews to provide descriptions as explanations for the recommendations they make [2, 5, 17, 34, 35, 37, 54, 57].

Zheng et al. [57] propose a factor model, named Multiple Similarities Collaborative Matrix Factorization (MSCMF), to address the problem of predicting new drug-target interactions. MSCMF assumes that a weighted linear combinations of a set of similarities can be approximated by the inner product of the corresponding two drugs' factor vectors, where the weights can be taken as the reasons why these two drugs are interacted according to the similarities. Barbieri et al. [2] propose a stochastic topic model, called WTFW (Who to Follow and Why), for the link prediction with explanations for user recommendation in social networks. WTFW not only predicts links, but also for each predicted link it decides whether it is a "topical" or a "social" link, and based on this decision it produces different types of explanations. However, the explanations, described by a set of Top-k features (for "topical" link) or common neighbors (for "social" link), are qualitative, while ours explanations are quantitative. Peng et al. [34] propose an unsupervised probabilistic model to learn the latent factors of users and products. Within the latent topic space, Peng et al. explain the review rating and review text. Ribeiro et al. [37] propose LIME, an explanation technique that explains the predictions of a classifier in an interpretable and faithful manner, by learning an interpretable model locally around the prediction. Zhang et al. [54] propose an Explicit Factor Model (EFM) to generate explainable recommendations, meanwhile keep a high prediction accuracy. He et al. [17] devise a generic algorithm for ranking on tripartite graphs — TriRank — and specialize it for personalized recommendation, by extracting aspects (i.e., the specific properties of items) from textual reviews. Ren et al. [35] propose a latent variable model, called social collaborative viewpoint regression (sCVR), for predicting item ratings based on user opinions and social relations. However, these works often depend on user review mining, which are unsuitable for cold-start cases where reviews of new user are unavailable.

Recently, our team has published two new works on explainable recommendation, including [18] and [43]. In [18], Hou et al. propose an Aspect-based Matrix Factorization (AMF) model, which is able to improve the accuracy of rating prediction by collaboratively decomposing the rating matrix with a fusion of the auxiliary information extracted from aspects. In AMF, the reason why a recommendation is made to a user is quantitatively explained by two proposed metrics, User Aspect Preference (UAP) and Item Aspect Quality (IAQ), which quantify user preference to a specific aspect and the review sentiment of item on an aspect, respectively. In [43], Shi et al. model the heterogeneous objects involved in a recommender system by a Heterogeneous Information Network (HIN), and propose a semantic path based personalized recommendation model, SemRec, to predict the rating scores of users on items, where the semantic path with personalized weight can be regarded as the explanation of recommendations. However, these works only focus on the explanation of recommendations and also depend on user review mining, which makes them only suitable for warm-start recommendation problems, too.

8 Conclusions

In this paper, we propose a unified framework, called Meta-Feature based Explainable Recommendation Framework (MERF), for both cold-start and warm-start recommendations

with quantitative explanations. Meta-feature reveals the preferential relationship between users and item features, due to which MERF can make recommendations for cold-start users/items and warm-start ones in a consistent fashion. We propose a Personalized Feature Preference (PFP) vector for MERF to characterize the different importance of item features to a user. By fusing PFP and meta-features, MERF can make a recommendation with a quantitative explanation. To improve the efficiency, we propose a parallel learning algorithm and an incremental updating algorithm for PFP. We conduct the experiments on three real world datasets, and compare MERF with seven baseline methods. The experimental results verify the effectiveness and efficiency of MERF and the statistical significance of the superiority of MERF compared with baselines.

Acknowledgments This work is supported by National Science Foundation of China through grant 61173099, and in part by NSF through grants CNS-1115234 and OISE-1129076.

References

1. Agarwal, V., Bharadwaj, K.K.: Recommending diverse friends in signed social networks based on adaptive soft consensus paradigm using variable length genetic algorithm. World Wide Web **21**(5), 1285–1321 (2018)
2. Barbieri, N., Bonchi, F., Manco, G.: Who to follow and why: Link prediction with explanations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, pp. 1266–1275. ACM (2014)
3. Barjasteh, I., Forsati, R., Ross, D., Esfahanian, A.H., Radha, H.: Cold-start recommendation with provable guarantees: A decoupled approach. IEEE Trans. Knowl. Data Eng. **28**(6), 1462–1474 (2016)
4. Beutel, A., Covington, P., Jain, S., Xu, C., Li, J., Gatto, V., Chi, E.H.: Latent cross: Making use of context in recurrent recommender systems. In: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM '18, pp. 46–54. ACM (2018)
5. Bilgic, M., Mooney, R.J.: Explaining recommendations: Satisfaction vs. promotion. In: Beyond Personalization Workshop, IUI, vol. 5, p. 153 (2005)
6. Casella, G., Berger, R.L.: Statistical Inference, vol. 2. Duxbury Pacific Grove, CA (2002)
7. Chen, J., Zhuang, F., Hong, X., Ao, X., Xie, X., He, Q.: Attention-driven factor model for explainable personalized recommendation. In: The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '18, pp. 909–912. ACM (2018)
8. Chou, S.Y., Yang, Y.H., Jang, J.S.R., Lin, Y.C.: Addressing cold start for next-song recommendation. In: Proceedings of the 10th ACM Conference on Recommender Systems, RecSys '16, pp. 115–118. ACM (2016)
9. Demsar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**(1), 1–30 (2006)
10. Diane, K., Teevan, J.: Implicit feedback for inferring user preference: A bibliography. ACM SIGIR Forum **37**(2), 18–288 (2003)
11. Dunn, O.J.: Multiple comparisons among means. J. Am. Stat. Assoc. **56**(293), 52–64 (1961)
12. Duricic, T., Lacic, E., Kowald, D., Lex, E.: Trust-based collaborative filtering: Tackling the cold start problem using regular equivalence. In: Proceedings of the 12th ACM Conference on Recommender Systems, RecSys '18, pp. 446–450. ACM (2018)
13. Elbadrawy, A., Karypis, G.: Feature-Based Similarity Models for Top-n Recommendation of New Items. Tech. rep., Department of Computer Science. University of Minnesota, Minneapolis (2013)
14. Funk, S.: Netflix update: Try this at home (2006)
15. Gantner, Z., Drumond, L., Freudenthaler, C., Rendle, S., Schmidt-Thieme, L.: Learning attribute-to-feature mappings for cold-start recommendations. In: 2010 IEEE International Conference on Data Mining, pp. 176–185 (2010)
16. Gunawardana, A., Meek, C.: A unified approach to building hybrid recommender systems. In: Proceedings of the Third ACM Conference on Recommender Systems, RecSys '09, pp. 117–124. ACM (2009)

17. He, X., Chen, T., Kan, M.Y., Chen, X.: Trirank: Review-aware explainable recommendation by modeling aspects. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, CIKM '15, pp. 1661–1670. ACM (2015)
18. Hou, Y., Yang, N., Wu, Y., Yu, P.S.: Explainable recommendation with fusion of aspect information. World Wide Web **22**(1), 221–240 (2019)
19. Hu, L., Cao, J., Xu, G., Cao, L., Gu, Z., Zhu, C.: Personalized recommendation via cross-domain triadic factorization. In: Proceedings of the 22nd International Conference on World Wide Web, WWW '13, pp. 595–606. ACM (2013)
20. Hyun, D., Park, C., Yang, M.C., Song, I., Lee, J.T., Yu, H.: Review sentiment-guided scalable deep recommender system. In: The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '18, pp. 965–968. ACM, New York (2018)
21. IMDb: http://www.imdb.com/ (2015)
22. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of ir techniques. ACM Trans. Inf. Syst. **20**(4), 422–446 (2002)
23. Jiang, M., Fang, Y., Xie, H., Chong, J., Meng, M.: User click prediction for personalized job recommendation. World Wide Web **22**(1), 325–345 (2019)
24. Karamanolakis, G., Cherian, K.R., Narayan, A.R., Yuan, J., Tang, D., Jebara, T.: Item recommendation with variational autoencoders and heterogeneous priors. In: Proceedings of the 3rd Workshop on Deep Learning for Recommender Systems, DLRS 2018, pp. 10–14. ACM (2018)
25. Kim, D., Park, C., Oh, J., Yu, H.: Deep hybrid recommender systems via exploiting document context and statistics of items. Inform. Sci. **417**, 72–87 (2017)
26. Lamar, R.N., Beswick, R.W.: Voice mail versus conventional channels: A cost minimization analysis of individuals' preferences. Acad. Manage. J. **33**(4), 901–816 (1990)
27. Lee, S., Song, S.i., Kahng, M., Lee, D., Lee, S.g.: Random walk based entity ranking on graph for multidimensional recommendation. In: Proceedings of the Fifth ACM Conference on Recommender Systems, RecSys '11, pp. 93–100. ACM (2011)
28. Li, C.Y., Lin, S.D.: Matching users and items across domains to improve the recommendation quality. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, pp. 801–810. ACM (2014)
29. Lu, K., Zhang, Y., Zhang, L., Wang, S.: Exploiting user and business attributes for personalized business recommendation. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '15, pp. 891–894. ACM (2015)
30. Ma, Y.: http://pan.baidu.com/s/1qXCaYvU/ (2016)
31. Netflix prize. http://www.netflixprize.com
32. Ning, X., Karypis, G.: Slim: Sparse linear methods for top-n recommender systems. In: 2011 IEEE International Conference on Data Mining, ICDM' 11, pp. 497–506. IEEE (2011)
33. Paterek, A.: Improving regularized singular value decomposition for collaborative filtering. In: Proceedings of KDD cup and workshop, pp. 5–8 (2007)
34. Peng, J., Zhai, Y., Qiu, J.: Learning latent factor from review text and rating for recommendation. In: 2015 7th International Conference on Modelling, Identification and Control (ICMIC), pp. 1–6 (2015)
35. Ren, Z., Liang, S., Li, P., Wang, S., de Rijke, M.: Social collaborative viewpoint regression with explainable recommendations. In: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM '17, pp. 485–494. ACM (2017)
36. Rendle, S.: Factorization machines. In: 2010 IEEE International Conference on Data Mining, ICDM' 10, pp. 995–1000. IEEE (2010)
37. Ribeiro, M.T., Singh, S., Guestrin, C.: "why should i trust you?": Explaining the predictions of any classifier. In: Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, pp. 1135–1144. ACM (2016)
38. Ronald, I.L., Davenport, J.M.: Approximations of the critical region of the fbietkan statistic. Commun. Statist.-Theory Methods **9**(6), 571–595 (1980)
39. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th International Conference on World Wide Web, WWW '01, pp. 285–295. ACM (2001)
40. Sedhain, S., Menon, A.K., Sanner, S., Xie, L., Braziunas, D.: Low-rank linear cold-start recommendation from social data. In: Proceedings of the 31st AAAI Conference on Artificial Intelligence, pp. 1502–1508 (2017)
41. Sharma, M., Zhou, J., Hu, J., Karypis, G.: Feature-based factorized bilinear similarity model for cold-start top-n item recommendation. In: Proceedings of the 2015 SIAM International Conference on Data Mining, pp. 190–198 (2015)

42. Shi, S., Zhang, M., Liu, Y., Ma, S.: Attention-based adaptive model to unify warm and cold starts recommendation. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM '18, pp. 127–136. ACM, New York (2018). https://doi.org/10.1145/3269206.3271710
43. Shi, C., Zhang, Z., Ji, Y., Wang, W., Yu, P.S., Shi, Z.: Semrec: a personalized semantic recommendation method based on weighted heterogeneous information networks. World Wide Web **22**(1), 153–184 (2019)
44. Silva, N., Carvalho, D., Pereira, A.C., Mourão, F., Rocha, L.: The pure cold-start problem: A deep study about how to conquer first-time users in recommendations domains. Inf. Syst. **80**, 1–12 (2019)
45. Tay, Y., Luu, A.T., Hui, S.C.: Multi-pointer co-attention networks for recommendation. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '18, pp. 2309–2318. ACM (2018)
46. Wang, Z., Liang, J., Li, R., Qian, Y.: An approach to cold-start link prediction: Establishing connections between non-topological and topological information. IEEE Trans. Knowl. Data Eng. **28**(11), 2857–2870 (2016)
47. Wu, C.Y., Ahmed, A., Beutel, A., Smola, A.J., Jing, H.: Recurrent recommender networks. In: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM '17, pp. 495–503. ACM (2017)
48. Xiang, L., Yuan, Q., Zhao, S., Chen, L., Zhang, X., Yang, Q., Sun, J.: Temporal recommendation on graphs via long- and short-term preference fusion. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '10, pp. 723–732. ACM (2010)
49. Xu, J., Yao, Y., Tong, H., Tao, X., Lu, J.: Rapare: A generic strategy for cold-start rating prediction problem. IEEE Trans. Knowl. Data Eng. **29**(6), 1296–1309 (2017)
50. Xu, Q., Shen, F., Liu, L., Shen, H.T.: Graphcar: Content-aware multimedia recommendation with graph autoencoder. In: The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '18, pp. 981–984. ACM (2018)
51. Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W.L., Leskovec, J.: Graph convolutional neural networks for web-scale recommender systems. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '18, pp. 974–983. ACM (2018)
52. Yu, Y., Gao, Y., Wang, H., Wang, R.: Joint user knowledge and matrix factorization for recommender systems. World Wide Web **21**(4), 1141–1163 (2018)
53. Zhang, M., Tang, J., Zhang, X., Xue, X.: Addressing cold start in recommender systems: A semi-supervised co-training algorithm. In: Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, pp. 73–82. ACM, New York (2014)
54. Zhang, Y., Lai, G., Zhang, M., Zhang, Y., Liu, Y., Ma, S.: Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In: Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, pp. 83–92. ACM (2014)
55. Zhao, Z., Cheng, Z., Hong, L., Chi, E.H.: Improving user topic interest profiles by behavior factorization. In: Proceedings of the 24th International Conference on World Wide Web, WWW '15, pp. 1406–1416. International World Wide Web Conferences Steering Committee (2015)
56. Zheng, V.W., Cao, B., Zheng, Y., Xie, X., Yang, Q.: Collaborative filtering meets mobile recommendation: A user-centered approach. In: Proceedings of the 24th AAAI Conference on Artificial Intelligence, pp. 236–241. AAAI (2010)
57. Zheng, X., Ding, H., Mamitsuka, H., Zhu, S.: Collaborative matrix factorization with multiple similarities for predicting drug-target interactions. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '13, pp. 1025–1033. ACM (2013)
58. Zhu, J., Zhang, J., Zhang, C., Wu, Q., Jia, Y., Zhou, B., Philip, S.Y.: Chrs: Cold start recommendation across multiple heterogeneous information networks. IEEE Access **51**, 5,283–15,299 (2017)