Embedding Factorization Models for Jointly Recommending Items and User Generated Lists

Da Cao[1,2], Liqiang Nie[3], Xiangnan He[4], Xiaochi Wei[5], Shunzhi Zhu[2,*], Tat-Seng Chua[4]
[1]Department of Automation, Xiamen University
[2]College of Computer and Information Engineering, Xiamen University of Technology
[3]School of Computer Science and Technology, Shandong University
[4]School of Computing, National University of Singapore
[5]School of Computer Science, Beijing Institute of Technology
{caoda0721, nieliqiang, xiangnanhe, xcwei.bit}@gmail.com; szzhu@xmut.edu.cn; dcscts@nus.edu.sg

ABSTRACT

Existing recommender algorithms mainly focused on recommending individual items by utilizing user-item interactions. However, little attention has been paid to recommend user generated lists (*e.g.,* playlists and booklists). On one hand, user generated lists contain rich signal about item co-occurrence, as items within a list are usually gathered based on a specific theme. On the other hand, a user's preference over a list also indicate her preference over items within the list. We believe that 1) if the rich relevance signal within user generated lists can be properly leveraged, an enhanced recommendation for individual items can be provided, and 2) if user-item and user-list interactions are properly utilized, and the relationship between a list and its contained items is discovered, the performance of user-item and user-list recommendations can be mutually reinforced.

Towards this end, we devise embedding factorization models, which extend traditional factorization method by incorporating item-item (item-item-list) co-occurrence with embedding-based algorithms. Specifically, we employ factorization model to capture users' preferences over items and lists, and utilize embedding-based models to discover the co-occurrence information among items and lists. The gap between the two types of models is bridged by sharing items' latent factors. Remarkably, our proposed framework is capable of solving the new-item cold-start problem, where items have never been consumed by users but exist in user generated lists. Overall performance comparisons and micro-level analyses demonstrate the promising performance of our proposed approaches.

CCS CONCEPTS

•Information systems → Recommender systems; •Human-centered computing → Collaborative filtering;

SIGIR'17, August 7-11, 2017, Shinjuku, Tokyo, Japan.

DOI: http://dx.doi.org/10.1145/3077136.3080779

Playlist A		
01	♥	Song A
02	♡	Song B
03	♥	Song C
04	♥	Song D
05	♡	Song E

Playlist B		
01	♡	Song F
02	♡	Song G
03	♥	Song H
04	♡	Song I
05	♡	Song J

Playlist C		
01	♥	Song C
02	♡	Song F
03	♥	Song A
04	♥	Song D
05	♥	Song H

Figure 1: The illustrations of 1) a user's preference over lists; 2) the user's preference over items within lists; and 3) relationships among items and lists.

KEYWORDS

Recommender Systems, Factorization Model, Embedding-based Model, Cold-start Problem, Co-occurrence Information

1 INTRODUCTION

Recommender systems have received great attention in recent years owing to their ability in mitigating the information overload problem. In the light of this, recommender systems have been widely applied in multiple domains such as music [6, 7], image [4, 5], restaurant [14], and mobile app [3, 21, 22]. Existing studies on recommender systems mainly focused on recommending individual and independent items to users, and arousing several derived research topics such as cold-start problem [2], implicit feedback [1, 13], context-aware [11, 27], deep learning [35], and efficient hashing [30, 37]. However, traditional recommender systems are specifically designed for optimizing user-item interactions and are not optimized for some complex circumstances such as recommending user generated lists, which are very popular in various real-world scenarios such as lists of songs created by listeners on Netease[1], lists of books created by readers on GoodReads[2], and lists of products created by shoppers on Amazon[3].

User generated lists are created by users and are exposed to the public by default. Beyond the traditional searching process for isolated items, the list service offers a new way for users to explore their desired group of items. Figure 1 illustrates a user's

* Corresponding author.
[1]https://music.163.com
[2]https://www.goodreads.com
[3]https://www.amazon.com

preference over lists and their contained items in the domain of music. The items are manually grouped into a list according to a specific theme, which indicates the co-occurrence feature among items. Meanwhile, as the list is composed of several items, the preference over a list somehow signal one's preference over the contained items. Therefore, if users' preferences over items and lists are properly utilized and the relationship between the list and its items is discovered, the recommendation performance of user-item and user-list can be mutually reinforced.

Despite its value and significance, jointly recommending user generated lists and their contained items remains in its infancy due to the following challenges: 1) Unlike the traditional item-independent view, items within a list are gathered together based on a common theme, which provides rich signal about the correlation of items. However, how to capture and model the co-occurrence relationship among items is a non-trivial task. 2) The cold-start problem is much heavier for item recommendation since there are numerous items only exist in lists but are never consumed by users (referred to as *new-item cold-start*). Thus how to leverage co-occurrence information among items from lists to alleviate the new-item cold-start problem is of great interest. 3) A user's preference over a list also manifests her preference over items within the list, and vice versa. Therefore, how to devise an unified framework to reinforce the recommendation performance of both user-item and user-list is a valuable research issue. In summary, to better serve users with a quality recommendation service, it is highly desirable to develop techniques that comprehensively consider the co-occurrence information among items, new-item cold-start problem, and joint recommendation of items and lists.

To tackle these problems, we devise embedding factorization models, which jointly factorize user-item (user-item-list) interaction matrix and item-item (item-item-list) co-occurrence matrix, where the co-occurrence matrix is derived from embedding-based algorithms. In the first stage, we utilize lists as side-information to harvest the item co-occurrence relationship. The co-occurrence information among items is obtained by using word embedding method, whereby items are correlated within a fixed-length window. Furthermore, user-item interaction matrix and item-item co-occurrence matrix are factorized at the same time. As a byproduct, the new-item cold-start problem can be solved. In the second stage, we incorporate the list as a word into the item-item co-occurrence matrix, and refer the newly created matrix as item-item-list co-occurrence matrix. Moreover, user-item interaction matrix, user-list interaction matrix, and item-item-list co-occurrence matrix are factorized within a unified framework. Thus, the recommendation performance of user-item and user-list can be mutually enhanced. By conducting experiments on our constructed real-world datasets, we demonstrate that our proposed approaches yield significant gains as compared with other state-of-the-art methods.

Our contributions are summarized as follows:

- We explore the promising yet challenging problem of recommending user generated lists and their contained items. To the best of our knowledge, this is the first work that attempts to solve this problem by reinforcing recommendation approach with embedding-based algorithms.
- We present embedding factorization models that jointly factorize user-item (user-item-list) interaction matrix and item-item (item-item-list) co-occurrence matrix. The new-item cold-start problem can be well addressed, and the user-item and user-list recommendation performance can be mutually enhanced.
- Extensive experiments performed on our self-collected datasets demonstrate the effectiveness of our proposed approaches. Meanwhile, we have released the datasets and our implementation to facilitate the research community for further exploration[4].

2 RELATED WORK

The development of recommender systems has long been divided into two directions, namely, rating prediction and item recommendation. Rating prediction on explicit feedback has dominated the research community in the early stage of recommender systems, primarily due to its simplicity on model optimization and evaluation (as only observer data need to be considered [15, 16, 35]). However, in real-world scenarios, interactions between users and items are always in an implicit form, such as purchasing products and listening music. Obviously, due to the one-class nature of implicit data, rating prediction is no longer suitable, and the research trends turn to item recommendation task. Various ranking algorithms were designed to model implicit datasets, among which Bayesian personalized ranking (BPR) [26] is a typical work. BPR optimizes ranking performance through a general pair-wise ranking function. Meanwhile, users' preferences over items are inferred by utilizing a negative sampling strategy. Other commonly used item recommendation frameworks include point-wise regression [1, 40] and graph-based models [12, 33].

The aforementioned item recommendation techniques have indeed improved the recommendation performance over individual items. However, in practice, users are often exposed to a set of items and the items are not independent. Along this line, a framework for recommending relevant and diverse items was proposed [29], which explicitly takes the coverage of user's interests into account. Another curiosity-based work was introduced [38], which combines relevance and curiosity in the recommendation process. Except for the work considering the correlation among items, recommending a list of several items instead of isolated items is another remarkable direction. In [41], the authors defined the concept of bundle recommendation, in which a bundle refers to a set of items that users consider or consume together under a specific circumstance (*e.g.,* limit total price, contextual influence, and product compatibility or consistency). The work of [28] aims to recommend a list of items to users by optimizing the list's click-trough rate. Although these work recommends a list of items to users, users' explicit preferences over lists are ignored. The closest work to ours is [24], in which users' previous interactions with both lists and individual items are considered simultaneously. The relationship between the list and its contained items is built based on a linear combination. Distinct from previous work, we treat a list as a sentence and items within the list as words. We then resort to word embedding algorithms to explore the list recommendation problem from such a new perspective.

[4] https://listrec.wixsite.com/efms

Word embedding models have gained great success in natural language processing tasks, where words or phrases are mapped to vectors of real numbers. As a seminal method for word embedding, word2vec [25] is able to capture a large number of precise syntactic and semantic word relationships. It has been shown that the word2vec (skip-gram) is equivalent to perform implicit matrix factorization [19]. As an extension to word2vec, paragraph2vec [18] is later introduced, in which paragraph refers to variable-length pieces of texts (*e.g.,* sentences, paragraphs, and documents). It is a two-step model, where word vectors and paragraph vectors are obtained in the first and second phase, respectively. Inspired by paragraph2vec, hierarchical neural language models [8] are proposed by designing joint training models to obtain word vectors and paragraph vectors simultaneously.

There have been some efforts to tailor word embedding algorithms for delivering effective recommendation. Grbovic *et al.* [10] developed prod2vec models, in which a product is treated as a word and the list of a user's consumed products is regarded as a sentence. The recommendation is conducted by computing the distance between the new product (word) and the list of user's consumed products (sentence). To further enhance the performance of prod2vec, Vasile *et al.* [32] proposed to inject item metadata into the prod2vec to regularize the item embedding. By utilizing the skip-gram word2vec model and a pair-wise ranking loss, Liu *et al.* [23] devised a two-step strategy to explore the context of locations for personalized location recommendation. A co-factorization model, CoFactor [20], was presented, which jointly decomposes the user-item interaction matrix and the item-item co-occurrence matrix by sharing items' latent factors. CoFactor is inspired by the word2vec model which can be interpreted as factorizing the word co-occurrence matrix. Word embedding algorithms have shown great ability in enhancing the recommendation performance for individual items. This work is orthogonal to the above mentioned work, as we exploit the embedding-based techniques to tackle the list recommendation task, which to our knowledge, has never been explored before.

3 PRELIMINARIES

We first introduce the Bayesian-based ranking model, BPR, which is designed for optimizing users' preferences over pair-wise samples (*i.e.,* positive sample against negative or missing sample). We then describe how the skip-gram word2vec model is translated into factorizing a word-context matrix. Finally, the hierarchical neural language model is introduced, which is devised for representing sentences and their contained words.

3.1 Bayesian Personalized Ranking

BPR is a latent factor-based algorithm [26]. In BPR, each user u is represented as a latent factor $\mathbf{w}_u$, and each item i is denoted as a latent factor $\mathbf{h}_i$. The predicted rating on item i by user u is represented as $\hat{x}_{ui}$ and is generated by the inner product of the corresponding user latent factor and item latent factor, *i.e.,* $\mathbf{w}_u \cdot \mathbf{h}_i$. The optimization criterion for BPR consists of a Bayesian analysis and the prior probability for the model parameters, which is represented as,

$$\begin{aligned}\mathcal{L}_{\text{BPR}} &= \ln p(\Theta| >_u) \\ &= \ln p(>_u |\Theta) p(\Theta) \\ &= \ln \prod_{(u,i,j)\in\mathcal{D}_B} \sigma(\hat{x}_{uij}) p(\Theta) \\ &= \sum_{(u,i,j)\in\mathcal{D}_B} \ln \sigma(\hat{x}_{uij}) + \ln p(\Theta) \\ &= \sum_{(u,i,j)\in\mathcal{D}_B} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta ||\Theta||, \end{aligned} \tag{1}$$

where $>_u$ denotes the desired latent preference structure for the user u; $\hat{x}_{uij} = \hat{x}_{ui} - \hat{x}_{uj}$; the observed subset $\mathcal{D}_B$ of $>_u$ is used as training data; $\sigma = \frac{1}{1+e^{-x}}$ is the logistic sigmoid function; Θ denotes all parameters (latent factors of users and items); and λ_Θ is the model specific regularization parameter. BPR is suitable for matrix factorization (MF) [16] and k-nearest-neighbor (KNN) [15]. When it comes to MF, the problem of predicting $\hat{x}_{ui}$ can be seen as the task of estimating a matrix $\mathbf{X} : U \times I$. The estimation of $\mathbf{X}$ is approximated by the matrix product of two low-rank matrices $\mathbf{W} : |U| \times k$ and $\mathbf{H} : |I| \times k$,

$$\hat{\mathbf{X}} := \mathbf{W}\mathbf{H}^t, \tag{2}$$

where k is the dimensionality of the approximation. Meanwhile, the prediction formula can be written as,

$$\hat{x}_{ui} =< \mathbf{w}_u, \mathbf{h}_i >= \sum_{f=1}^{k} w_{uf} \cdot h_{if}. \tag{3}$$

To optimize the criteria of BPR, a stochastic gradient descent algorithm, L_{EARN}BPR, is proposed which is based on bootstrap sampling of training triples. The model parameters for matrix factorization are $\Theta = (\mathbf{W}, \mathbf{H})$. The optimization with L_{EARN}BPR only lies on $\frac{\partial \hat{x}_{uij}}{\partial \theta}$.

3.2 Word Embedding as Matrix Factorization

The skip-gram word2vec model [25] is a simplified neural language model without any non-linear hidden layers. A log-linear classifier is used to predict the surrounding words within a certain distance based on the current one. To be precise, the object of the skip-gram model is to maximize the log-likelihood,

$$\mathcal{L}_{\text{SG}} = \sum_{t=1}^{T} \log \mathbb{P}(w_{t-c} : w_{t+c} | w_t), \tag{4}$$

where w_t is the t^{th} word in the sequence, and $w_{t-c} : w_{t+c}$ represents successive words $(w_{t-c}, w_{t-c+1}, ..., w_{t+c})$ that act as the context to the word w_t. Several sophisticated implementations are designed for the skip-gram model, such as negative sampling and hierarchical softmax.

Levy and Goldberg [19] demonstrated that the skip-gram with negative-sampling word2vec model is implicitly factorizing a word-context matrix, whose cells are the pointwise mutual information (PMI) of the respective word and context pairs, shifted by a global constant. Given a training corpus $\mathcal{D}$ and a word i, several words are selected as the context words for word i according to a specific strategy (*e.g.,* the neighboring words falling in the fix-sized windows

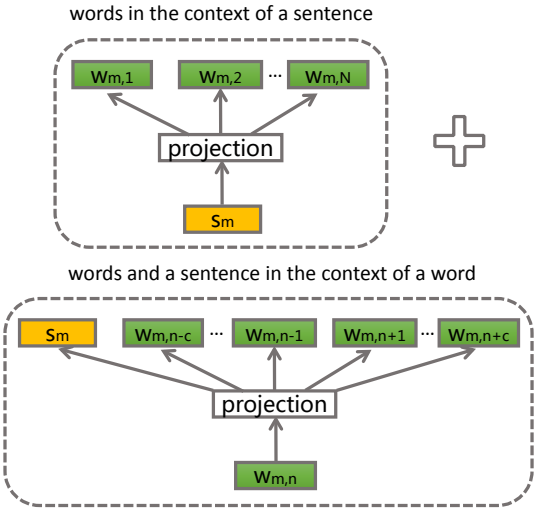


Figure 2: The visualization of the embedding model for sentences and their contained words.

centered at word i). The number of times that a context word j appears in the word i's context is denoted as $\#(i,j)$. PMI between a word i and its context j is defined as,

$$\text{PMI}(i,j) = \log \frac{\#(i,j) \cdot \mathcal{D}}{\#(i) \cdot \#(j)}, \tag{5}$$

where $\#(i) = \sum_j \#(i,j)$, $\#(j) = \sum_i \#(i,j)$, and $\mathcal{D} = \sum_{ij} \#(i,j)$. Levy and Goldberg [19] further proposed to perform word embedding by spectral dimensionality reduction (*e.g.,* singular value decomposition) on the (sparse) shifted positive PMI (SPPMI) matrix,

$$\text{SPPMI}(i,j) = \max\{\text{PMI}(i,j) - \log h, 0\}, \tag{6}$$

where h is a hyperparameter that controls the density of the SPPMI matrix.

3.3 Embedding Model for Sentences

As an extension of the skip-gram word2vec model in Eq. (4), Le and Mikolov [18] proposed the paragraph2vec model to obtain the paragraph representation, which utilizes a two-phase strategy. Hierarchical neural language models [8, 10] were further presented to jointly represent sentences and their contained words. The training dataset is derived from the set of sentences $\mathcal{G}$, which comprised of sentences s_m and their contained sequential words $s_m = (w_{m1}, w_{m2}, ..., w_{mN})$, where N denotes the sentence length. As illustrated in [8], both skip-gram and continuous bag-of-word models can be used in any level of the hierarchy. We thus resort to utilize skip-gram in all levels of the hierarchy in order to make it consistent with Section 3.2. The architecture of such model is illustrated in Figure 2. The objective of the hierarchical model is to maximize the log-likelihood of the streaming data,

$$\mathcal{L}_{\text{HM}} = \sum_{g \in \mathcal{G}} \Big(\sum_{w_{mn} \in s_m} \log \mathbb{P}(w_{m,n-c} : w_{m,n+c}, s_m | w_{mn}) + \sum_{s_m \in g} (\log \mathbb{P}(w_{m1} : w_{mN} | s_m)), \tag{7}$$

where c is the length of the training context for word sequences.

4 OUR PROPOSED APPROACHES

In this section, we present our embedding factorization models for jointly recommending user generated lists and their contained items. In the first stage, user generated lists are employed as side-information to correlate items within them. Meanwhile, the new-item cold-start problem can be solved, in which items exist in lists but are new to users. In the second stage, both user-item interactions and user-list interactions are considered, user generated lists and their contained items are recommended simultaneously.

4.1 Utilizing Lists as Side-Information

4.1.1 Exploring the Item Co-occurrence. Given the sparse user-item interaction matrix $\mathbf{X} \in \mathbb{R}^{U \times I}$ from U users and I items, where I items are from L lists, BPR decomposes it into the product of user and item latent factors. User generated lists are considered as side-information, which means users' preferences over lists are ignored in this stage but lists are utilized to correlate items within them. Let $\mathbf{M} \in \mathbb{R}_+^{I \times J}$ be the co-occurrence SPPMI matrix, where I represents the number of items in lists and J represents the number of context items[5]. The lists are divided into shorter segments of sequences according to the fixed-length window size c. $m_{ij} \in \mathbf{M}$ is computed according to Eq. (5) and Eq. (6). $\mathbf{M}$ is factorized into two low-rank matrices $\mathbf{H} : |I| \times k$ and $\mathbf{C} : |J| \times k$,

$$\hat{\mathbf{M}} := \mathbf{H}\mathbf{C}^t, \tag{8}$$

where k is the dimensionality of the approximation. The task of the recommendation is to provide the user with a personalized total ranking $>_u$ of all items I, sorted according to the likelihood that the user will be interested in each of them.

Both BPR and word2vec's equivalent matrix factorization are latent factor-based methods. The main difference is that the former factorizes user-item interaction matrix to capture users' preferences over items, while the latter factorizes item-item co-occurrence matrix to obtain the co-occurrence relationship among items. Items' latent factors are shared across these two models. Therefore, we propose a joint learning model to seamlessly sew them up,

$$\mathcal{L}_{\text{EFM-Side}} = \sum_{(u,i,j) \in \mathcal{D}_T} \ln \sigma(\hat{x}_{uij}) + \lambda_s \sum_{(i,j,t) \in \mathcal{D}_S} \ln \sigma(\hat{m}_{ijt}) - \lambda_\Theta ||\Theta||, \tag{9}$$

where $\mathcal{D}_T$ is the training corpus for the user-item entry $\hat{x}_{ui}$; $\mathcal{D}_S$ is the training corpus for the item-item entry $\hat{m}_{ij}$; λ_s balances the performance between BPR and word2vec's equivalent matrix factorization. This is the embedding factorization model by utilizing lists as side-information (denoted as EFM-Side). In fact, several co-training models (*e.g.,* collective matrix factorization [31] and multi-relational matrix factorization [17]) have been proposed, which co-factorize multiple matrices in the context of relational learning. We can regard EPM-Side as a special case of MR-BPR [17], which builds the co-occurrence relationship among items.

[5] In fact, the set of items in lists and the set of context items are the same.

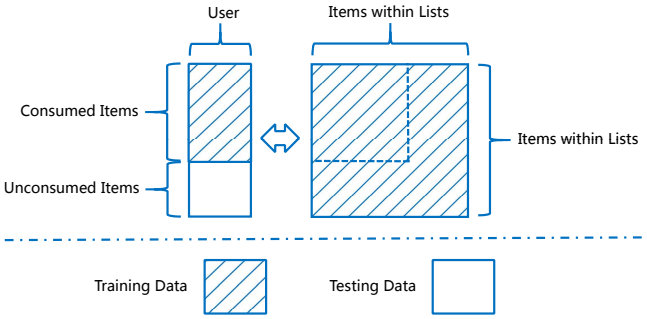


Figure 3: The illustration of the new-item cold-start problem, where cold-start items only exist in lists and are never consumed by users.

4.1.2 New-Item Cold-Start Problem. Apart from utilizing list as side-information to boost recommendation performance, we expect the new-item cold-start problem to be solved as well. In the new-item cold-start scenario, the task is to recommend the items that only exist in lists but are totally fresh to users. Figure 3 visualizes the new-item cold-start problem. The latent factors for users' consumed items are finely learnt from user-item interactions. For unconsumed items, their latent factors are adjusted by factorizing item-item co-occurrence matrix. Therefore, users' preferences over unconsumed items are estimated from the inner products of users' latent factors and the unconsumed items' latent factors. It is worth noting that in addition to the new-item cold-start problem, EFM-Side can also well handle warm-start scenarios where the initial few user-item interactions become available.

4.2 Jointly Recommending Items and Lists

Suppose we have U users, I items, and L lists, where I items are from L lists. Beside the sparse user-item interaction matrix $\mathbf{X} \in \mathbb{R}^{U \times I}$, the sparse user-list interaction matrix $\mathbf{Y} \in \mathbb{R}^{U \times L}$ is also available. As we know, the sentence embedding model is an extension to skip-gram word2vec, word2vec's equivalent matrix factorization can thus naturally expand to the embedding model for sentences. In Figure 2, the sentence is regarded as a word token to be embedded into the word2vec model. Regarding the concept of window size, the lists are further divided into shorter segments of sequences with the length of c. The sentence is appeared in the context of each word within it, and each word within the sentence is also appeared in the context of the sentence. Therefore, the co-occurrence SPPMI matrix $\mathbf{M} \in \mathbb{R}_+^{I \times J}$ is further extended to $\mathbf{R} \in \mathbb{R}_+^{|I'| \times |J'|}$, where $|I'|$ represents the number of items and lists, and $|J'|$ represents the number of context items and lists. Obviously, $|I'| = |J'| = |I| + |J|$. The elements in $\mathbf{R}$ are computed according to Eq. (5) and Eq. (6). Hence, we propose a joint learning model to enhance the recommendation performance of both item and list,

$$\mathcal{L}_{\text{EFM-Joint}} = \sum_{(u,i,j)\in\mathcal{D}_T} \ln \sigma(\hat{x}_{uij}) + \sum_{(u,i,j)\in\mathcal{D}_L} \ln \sigma(\hat{y}_{uij}) + \lambda_r \sum_{(i,j,t)\in\mathcal{D}_R} \ln \sigma(\hat{r}_{ijt}) - \lambda_{\Theta}||\Theta||, \tag{10}$$

Algorithm 1: The Optimization for EFM-Side

```
Random initialize Θ;
repeat
    // Factorize user-item interaction matrix
    repeat
        Draw (w_u, h_i, h_j) from D_T;
        Θ ← Θ + μ ( e^{-x̂_uij} / (1+e^{-x̂_uij}) · ∂x̂_uij/∂Θ + λ_Θ · Θ );
    until convergence;
    // Factorize item-item co-occurrence matrix
    repeat
        Draw (h_i, c_j, c_t) from D_S;
        Θ ← Θ + μ ( λ_s e^{-m̂_ijt} / (1+e^{-m̂_ijt}) · ∂m̂_ijt/∂Θ + λ_Θ · Θ );
    until convergence;
until convergence or max-iteration has been reached;
```

where $\mathcal{D}_T$ is the training corpus for the user-item entry $\hat{x}_{ui}$; $\mathcal{D}_L$ is the training corpus for the user-list entry $\hat{y}_{ui}$; $\mathcal{D}_R$ is the training corpus for the item-item-list entry $\hat{r}_{ij}$; and λ_r balances the performance between BPR and the embedding model for sentences' equivalent factorization model. This is the embedding factorization model which jointly recommends lists and their contained items (denoted as EFM-Joint).

4.3 Solutions

4.3.1 Optimization for EFM-Side. The optimization procedure for the loss function in Eq. (9) can be realized via the stochastic gradient descent (SGD) strategy. A sophisticated SGD strategy algorithm, called LEARNBPR, is proposed [26], which only considers user-item interactions. As an extension to BPR, EFM-Side further takes the item co-occurrence into account. We inherit the multi-relational SGD strategy proposed in [17] to realize our designed framework. Specifically, the optimization procedure is conducted alternatively with respect to $\mathcal{D}_T$ and $\mathcal{D}_S$. At each iteration of $\mathcal{D}_T$ and $\mathcal{D}_S$, a training instance is randomly sampled, and a gradient descent step for all related parameters regarding the loss of the training instance is performed. Algorithm 1 details the procedure. The derivative of the loss function presented in Eq. (9) is:

$$\frac{\partial \mathcal{L}_{\text{EFM-Side}}(\hat{x}_{uij})}{\partial \Theta} = \frac{-e^{-\hat{x}_{uij}}}{1+e^{-\hat{x}_{uij}}} \cdot \frac{\partial \hat{x}_{uij}}{\partial \Theta} - \lambda_{\Theta} \cdot \Theta, \tag{11}$$

$$\frac{\partial \mathcal{L}_{\text{EFM-Side}}(\hat{m}_{ijt})}{\partial \Theta} = \lambda_s \frac{-e^{-\hat{m}_{ijt}}}{1+e^{-\hat{m}_{ijt}}} \cdot \frac{\partial \hat{m}_{ijt}}{\partial \Theta} - \lambda_{\Theta} \cdot \Theta. \tag{12}$$

The derivatives of $\hat{x}_{uij}$ and $\hat{m}_{ijt}$ are,

$$\frac{\partial \hat{x}_{uij}}{\partial \Theta} = \begin{cases} \mathbf{h}_{if} - \mathbf{h}_{jf} & if \quad \theta = \mathbf{w}_{uf}, \\ \mathbf{w}_{uf} & if \quad \theta = \mathbf{h}_{if}, \\ -\mathbf{w}_{uf} & if \quad \theta = \mathbf{h}_{jf}, \\ 0 & else \end{cases} \tag{13}$$

$$\frac{\partial \hat{m}_{ijt}}{\partial \Theta} = \begin{cases} \mathbf{c}_{jf} - \mathbf{c}_{tf} & if \quad \theta = \mathbf{h}_{if}, \\ \mathbf{h}_{if} & if \quad \theta = \mathbf{c}_{jf}, \\ -\mathbf{h}_{if} & if \quad \theta = \mathbf{c}_{tf}, \\ 0 & else \end{cases} \tag{14}$$

Table 1: Statistics of the evaluation datasets.

Dataset	User-song interaction#	User-list interaction#	User#	Song#	Playlist#	User-song density	User-list density
User-Song	1,128,065	0	18,528	123,628	22,864	0.05%	0
User-Song-Playlist	1,128,065	528,128	18,528	123,628	22,864	0.05%	0.12%

Algorithm 2: The Optimization for EFM-Joint

Random initialize Θ;
repeat
  // Factorize user-item interaction matrix
  repeat
    Draw $(\mathbf{w}_u, \mathbf{h}_i, \mathbf{h}_j)$ from $\mathcal{D}_T$;
    $\Theta \leftarrow \Theta + \mu\left(\frac{e^{-\hat{x}_{uij}}}{1+e^{-\hat{x}_{uij}}} \cdot \frac{\partial \hat{x}_{uij}}{\partial \Theta} + \lambda_\Theta \cdot \Theta\right)$;
  until *convergence*;
  // Factorize user-list interaction matrix
  repeat
    Draw $(\mathbf{w}_u, \mathbf{h}_i, \mathbf{h}_j)$ from $\mathcal{D}_L$;
    $\Theta \leftarrow \Theta + \mu\left(\frac{e^{-\hat{y}_{uij}}}{1+e^{-\hat{y}_{uij}}} \cdot \frac{\partial \hat{y}_{uij}}{\partial \Theta} + \lambda_\Theta \cdot \Theta\right)$;
  until *convergence*;
  // Factorize item-item-list co-occurrence matrix
  repeat
    Draw $(\mathbf{h}_i, \mathbf{c}_j, \mathbf{c}_t)$ from $\mathcal{D}_R$;
    $\Theta \leftarrow \Theta + \mu\left(\lambda_r \frac{e^{-\hat{r}_{ijt}}}{1+e^{-\hat{r}_{ijt}}} \cdot \frac{\partial \hat{r}_{ijt}}{\partial \Theta} + \lambda_\Theta \cdot \Theta\right)$;
  until *convergence*;
until *convergence or max-iteration has been reached*;

where f denotes the f^{th} latent feature of the corresponding latent factor. We alternatively factorize user-item interaction matrix and item-item co-occurrence matrix until convergence or the maximum number of iterations is reached. The optimization procedure applies to the new-item cold-start problem as well.

4.3.2 Optimization for EFM-Joint. The optimization procedure for EFM-Joint is similar to the one illustrated in Section 4.3.1. The main difference is that the user-list interaction matrix is considered and the co-occurrence matrix is reconstructed which takes the list into account. We thus still resort to conduct SGD optimization alternatively with respect to $\mathcal{D}_T$, $\mathcal{D}_L$, and $\mathcal{D}_R$. Algorithm 2 details the detailed procedure. The derivative of the loss function presented in Eq. (10) is,

$$\frac{\partial \mathcal{L}_{\text{EFM-Joint}}(\hat{x}_{uij})}{\partial \Theta} = \frac{-e^{-\hat{x}_{uij}}}{1+e^{-\hat{x}_{uij}}} \cdot \frac{\partial \hat{x}_{uij}}{\partial \Theta} - \lambda_\Theta \cdot \Theta, \tag{15}$$

$$\frac{\partial \mathcal{L}_{\text{EFM-Joint}}(\hat{y}_{uij})}{\partial \Theta} = \frac{-e^{-\hat{y}_{uij}}}{1+e^{-\hat{y}_{uij}}} \cdot \frac{\partial \hat{y}_{uij}}{\partial \Theta} - \lambda_\Theta \cdot \Theta, \tag{16}$$

$$\frac{\partial \mathcal{L}_{\text{EFM-Joint}}(\hat{r}_{ijt})}{\partial \Theta} = \lambda_r \frac{-e^{-\hat{r}_{ijt}}}{1+e^{-\hat{r}_{ijt}}} \cdot \frac{\partial \hat{r}_{ijt}}{\partial \Theta} - \lambda_\Theta \cdot \Theta. \tag{17}$$

The derivatives of $\hat{x}_{uij}$, $\hat{y}_{uij}$, and $\hat{r}_{ijt}$ are,

$$\frac{\partial \hat{x}_{uij}}{\partial \Theta} = \begin{cases} \mathbf{h}_{if} - \mathbf{h}_{jf} & if \quad \theta = \mathbf{w}_{uf}, \\ \mathbf{w}_{uf} & if \quad \theta = \mathbf{h}_{if}, \\ -\mathbf{w}_{uf} & if \quad \theta = \mathbf{h}_{jf}, \\ 0 & else \end{cases} \tag{18}$$

$$\frac{\partial \hat{y}_{uij}}{\partial \Theta} = \begin{cases} \mathbf{h}_{if} - \mathbf{h}_{jf} & if \quad \theta = \mathbf{w}_{uf}, \\ \mathbf{w}_{uf} & if \quad \theta = \mathbf{h}_{if}, \\ -\mathbf{w}_{uf} & if \quad \theta = \mathbf{h}_{jf}, \\ 0 & else \end{cases} \tag{19}$$

$$\frac{\partial \hat{r}_{ijt}}{\partial \Theta} = \begin{cases} \mathbf{c}_{jf} - \mathbf{c}_{tf} & if \quad \theta = \mathbf{h}_{if}, \\ \mathbf{h}_{if} & if \quad \theta = \mathbf{c}_{jf}, \\ -\mathbf{h}_{if} & if \quad \theta = \mathbf{c}_{tf}, \\ 0 & else \end{cases} \tag{20}$$

where f also denotes the f^{th} latent feature of the corresponding latent factor. The factorizations for user-item interaction matrix, user-list interaction matrix, and item-item-list co-occurrence matrix are alternatively conducted until convergence or max-iteration has been reached.

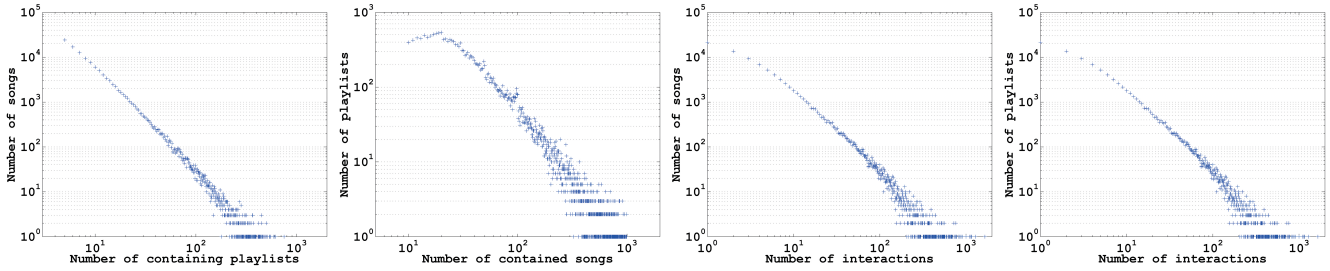
5 EXPERIMENTS

In this section, we conduct extensive experiments on our self-collected datasets to answer the following five research questions:

- **RQ1** How does our designed EFM-Side approach perform as compared with other state-of-the-art competitors?
- **RQ2** How does EFM-Side perform in handling the new-item cold-start problem?
- **RQ3** Does EFM-Side consistently outperform other algorithms with respect to items with differenct scale of ratings?
- **RQ4** How is the recommendation performance of user-item and user-list under the EFM-Joint framework?
- **RQ5** Are the items within a list equally important? Is EFM-Joint able to find the most representative item within a list?

5.1 Experimental Settings

5.1.1 Datasets Construction. We constructed the dataset by crawling data from Netease Cloud Music, which enables consumers to select their desired independent songs or user generated playlists via entering keywords or browsing from genres. We further processed the dataset by retaining playlists possessing at least 10 songs, songs appearing at least in 5 playlists, and users consuming at least 10 songs and 10 playlists. Based on these criteria, we ultimately obtained 18,528 users, 123,628 songs, 22,864 playlists, 1,128,065 user-song interactions, and 528,128 user-playlist interactions. In the experiments of evaluating EFM-Side, user-playlist interactions were erased and we denoted this dataset as User-Song. Meanwhile, in the experiments on EFM-Joint, user-playlist interactions were considered and we denoted this dataset as User-Song-Playlist. Statistics of these two datasets are summarized in Table 1.



(a) Song-playlist. **(b) Playlist-song.** **(c) Song-interaction.** **(d) Playlist-interaction.**

Figure 4: Dataset statistics w.r.t. song and playlist.

To gain insights into the data with respect to songs and playlists, we performed some statistical analysis. We plotted the song and playlist distributions with respect to the number of containing playlists and the number of contained songs in Figure 4a and 4b (log-log plot), respectively. As we can see, both songs and playlists show a long-tail distribution — most songs appeared in few playlists, and only a small proportion of playlists have many songs. The distributions of songs and playlists with different scales of interactions are revealed in Figure 4c and 4d, which also show a long-tail distribution. This is consistent with most recommendation benchmarks, such as the Nexflix [15] and Yelp [14] datasets, and highlights the sparsity challenge faced by recommender systems.

In our experiments, we used two different strategies to generate the training, validation, and testing sets for different evaluation settings. 1) In the overall performance comparisons for EFM-Side and EFM-Joint, the original User-Song and User-Song-Playlist datasets were divided into three disjoint sets respectively, with 80%, 10%, and 10% randomly selected interactions for training, validation and testing, respectively. 2) In the new-item cold-start problem handling, 80% of songs were randomly selected and their corresponding user-song interactions were utilized for training, 10% of songs were randomly selected and their user-song interactions were used for validation, and the remaining 10% of songs and their user-song interactions were used for testing.

5.1.2 Evaluation Metrics. As practical recommender systems usually generate a ranked list of items for a given user, we evaluate the ranking performance. AUC [26], the area under the ROC curve, is a commonly used metric for evaluating the quality of a ranking list. Let the disjoint training and testing sets be $\mathcal{S}_{\text{train}}$ and $\mathcal{S}_{\text{test}}$, respectively. The average AUC can be computed as,

$$\text{AUC} = \frac{1}{|U|}\sum_{u}\frac{1}{|E(u)|}\sum_{(i,j)\in E(u)}\delta(\hat{x}_{ui} > \hat{x}_{uj}), \tag{21}$$

where $\delta(\hat{x}_{ui} > \hat{x}_{uj})$ is an indicator function which returns 1 if $\hat{x}_{ui} > \hat{x}_{uj}$ is true, and 0 otherwise. The evaluation pair for each user u is,

$$E(u) := \{(i,j)|(u,i) \in \mathcal{S}_{\text{test}} \wedge (u,j) \notin (\mathcal{S}_{\text{test}} \cup \mathcal{S}_{\text{train}})\}. \tag{22}$$

A higher value of AUC indicates better performance for ranking performance. The floor of AUC from random guess is 0.5 and the best result is 1.

5.1.3 Baseline Methods. To justify the effectiveness of our proposed EFM-Side and EFM-Joint methods, we compared with the following state-of-the-art algorithms.

BPR [26]. This is a sampling-based algorithm that optimizes the pair-wise ranking between observed instances and sampled negative instances. It optimizes parameters by SGD, and is used in the experiments for benchmarking the overall performance of both EFM-Side and EFM-Joint.

BPR-map [9]. This is a two-step model, where the latent factors for the entities in the auxiliary domain are obtained in the first step and the latent factors for other entities in the target domain are acquired in the second step. It is mainly designed for mitigating the cold-start issue in the target domain. It would be used in the experiment for new-item cold-start problem.

LIRE [24]. This method considers users' previous interactions with both individual items and user generated lists. It weights items within lists based on both position of items and personalized list consumption pattern. It also applies the BPR framework for optimization, and is a strong competitor in jointly recommending lists and their contained items.

CoFactor [20]. This is a joint learning model that combines recommendation algorithm with word embedding model. The co-occurrence relationships among items (lists) are discovered from users' consumed sequential items (lists). It would be deployed in overall performance comparisons for both EFM-Side and EFM-Joint.

All hyper-parameters and learning rates of aforementioned approaches were carefully tuned on the validation set by grid search. We repeated every experiment for 5 times to report the average results. We also conducted the paired two-sample t-test based on the 5 times experiment results. We used Python for our algorithm implementations. All the experiments were conducted over a server equipped with Intel(R) Core(TM) i7-6700 CPU at 4.00 GHz on 32G RAM, 8 cores and 64-bit Windows 7 operating system.

5.2 Individual Items Recommendation (RQ1)

To demonstrate the overall effectiveness of our proposed EFM-Side as introduced in Section 5.1.3, we compared the EFM-Side with state-of-the-art item recommendation approaches BRP and CoFactor. Users' interactions with lists were ignored in this stage. EFM-Side acquires the co-occurrence relationships among items from user generated lists, while CoFactor obtains them from user consumed item sequences.

Table 2: Overall performance comparison under the EFM-Side framework.

Methods	k=10		k=20	
	AUC	p-value	AUC	p-value
BPR	0.9101 ± 0.002	3.49*e*-10	0.9149 ± 0.003	2.86*e*-10
CoFactor	0.9226 ± 0.003	5.09*e*-09	0.9221 ± 0.004	9.95*e*-10
EFM-Side	**0.9357 ± 0.004**	–	**0.9418 ± 0.003**	–

Table 3: Models comparison in handling the new-item cold-start problem.

Methods	k=10		k=20	
	AUC	p-value	AUC	p-value
Random	0.4902 ± 0.002	1.14*e*-14	0.5019 ± 0.002	8.35*e*-15
BPR-map	0.7729 ± 0.003	8.30*e*-12	0.7777 ± 0.003	2.25*e*-12
EFM-Side	**0.8381 ± 0.004**	–	**0.8680 ± 0.003**	–

Experimental results are shown in Table 2. We have the following observations: 1) Our EFM-Side achieves the AUC of 0.9357 and 0.9418 when $k = 10$ and $k = 20$, respectively. It shows substatial improvements over BPR, CoFactor of 2.81%, 1.42% when $k = 10$, and 2.94%, 2.14% when $k = 20$. All the p-values between our model and each of the baselines are much smaller than 0.05, indicating that the improvements are statistically significant. This validates that the accuracy of EFM-Side can be strengthened through harvesting co-occurrence relationships among items from user generated lists. 2) The performance of CoFactor is superior to that of BPR. CoFactor does not take advantage of any extra information, but the co-occurrence relationships among items can still be uncovered from user consumed item sequences. The cost is some additional computation. 3) EFM-Side achieves preferable results as compared with CoFactor, which illustrates that the co-occurrence relationships among items generated from user generated lists are more useful as compared with that of user consumed item lists.

5.3 New-Item Cold-Start Problem (RQ2)

As introduced in Section 4.1.2, the new-item cold-start problem refers to recommend items that are never consumed by users but exist in user generated lists. We compared the performance of EFM-Side with that of 1) Random and 2) BPR-map. Random is the worst result for ranking performance, which randomly selects items for recommendation. For the new-item cold-start scenario, if we know nothing about the new items, the random guess is the most reasonable results. BPR-map utilizes a two-step strategy to cope with the cold-start problem. In the first step of BPR-map, the latent factors of items are obtained by factorizing the item-item co-occurrence matrix. In the second step, user personalized ranking are optimized by fixing item latent factors, which are obtained in the first stage, and adjusting user latent factors.

Results are displayed in Table 3. We observed: 1) EFM-Side achieves the AUC of 0.8381 and 0.8680 when $k = 10$ and $k = 20$, respectively, which gains improvements over Random, BPR-map at 70.97%, 8.43% when $k = 10$, and 72.94%, 11.61% when $k = 20$. The paired two-sample t-test also supports the conclusion of significant improvements. Experimental results demonstrate the effectiveness of EFM-Side in handling the new-item cold-start problem. 2) The random guess for the new-item cold-start scenario is around 0.5 with respect to AUC. In fact, the performance of both BPR-map and EFM-Side significantly exceeds the bottom line, which illustrates the effectiveness of these methods in handling the new-item cold-start problem. 3) EFM-Side consistently outperforms BPR-map with respect to different sizes of latent factors. The main difference between BPR-map and EFM-Side is that BPR-map utilizes a two-step process while EFM-Side employs a joint learning strategy. The latent factors separately obtained from the factorization of item-item co-occurrence matrix and the factorization of user-item interaction matrix belong to two different spaces. EFM-Side connects these two different semantic spaces by sharing common item latent factors, which obviously outperforms the suboptimal two-step approach.

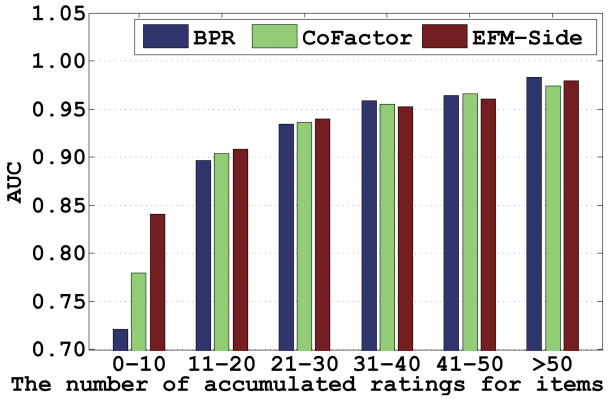


Figure 5: Micro-analysis w.r.t. items with different scale of accumulated ratings.

5.4 Performance Analysis w.r.t. Items (RQ3)

The data sparsity problem is extremely serious as revealed in Figure 4c. Moreover, we know EFM-Side overall outperforms other algorithms from the results illustrated in Section 5.2, but we are not sure whether EFM-Side is consistently superior to other competitors with respect to items with different scales of accumulated ratings. In order to answer this question, we further disposed the results obtained in Section 5.2 by selecting items whose number of training ratings is located in specific range (*i.e.*, 1-10, 11-20, ..., > 50).

The performance of various methods is shown in Figure 5. We have the following observations: 1) EFM-Side outperforms BPR and CoFactor for sparse items (*i.e.,* accumulated ratings of 1-10, 11-20, and 21-30). EFM-Side considers the co-occurrence information among items, which has a great effect on alleviating the data sparsity problem when the item has only a few ratings. 2) With the increasing of the number of training ratings for items, the performance of BPR, CoFactor, and EFM-Side is getting closer. This is because that when enough ratings are obtained for the items, the latent factors of items can be well learnt even without co-occurrence information among items.

Table 4: Overall performance comparison under the EFM-Joint framework w.r.t. item recommendation.

Methods	k=10		k=20	
	AUC	p-value	AUC	p-value
BPR	0.9104 ± 0.003	4.30e-10	0.9175 ± 0.002	3.49e-10
LIRE	0.9194 ± 0.003	2.73e-09	0.9218 ± 0.004	7.28e-10
CoFactor	0.9231 ± 0.004	8.28e-09	0.9245 ± 0.003	1.25e-09
EFM-Joint	**0.9347 ± 0.003**	–	**0.9431 ± 0.003**	–

Table 5: Overall performance comparison under the EFM-Joint framework w.r.t. list recommendation.

Methods	k=10		k=20	
	AUC	p-value	AUC	p-value
BPR	0.8593 ± 0.002	9.56e-10	0.8729 ± 0.003	2.07e-09
LIRE	0.8675 ± 0.004	8.00e-09	0.8818 ± 0.003	4.73e-08
CoFactor	0.8605 ± 0.003	1.22e-09	0.8738 ± 0.004	2.59e-09
EFM-Joint	**0.8792 ± 0.003**	–	**0.8893 ± 0.004**	–

5.5 Jointly Recommend Items and Lists (RQ4)

In the EFM-Joint framework, the recommendation performance of items and lists can be mutually reinforced. We compared the performance of EFM-Joint with other state-of-the-art algorithms: 1) BPR; 2) LIRE; and 3) CoFactor. LIRE treats the list as a combination of items, and user preferences over items and lists are mutually reinforced. CoFactor discovers the co-occurrence among items and lists from users' interactions with both items and lists.

The comparison results are illustrated in Table 4 and Table 5. We can observe that: 1) Regarding to the item recommendation performance, EFM-Joint achieves the AUC of 0.9347 and 0.9431 when $k = 10$ and $k = 20$, which gains improvements over BPR, LIRE, CoFactor at 2.67%, 1.66%, 1.26% when $k = 10$, and 2.79%, 2.31%, 2.01% when $k = 20$. When it comes to the list recommendation, EFM-Joint obtains the AUC of 0.8792 and 0.8893 when $k = 10$ and $k = 20$, which shows improvements over BPR, LIRE, CoFactor at 2.32%, 1.35%, 2.17% when $k = 10$, and 1.88%, 0.85%, 1.77% when $k = 20$. The improvements are statistically significant. EFM-Joint consistently outperforms other competitors in the recommendation performance of both item and list, which demonstrates the effectiveness of our proposed joint learning framework. 2) LIRE beats CoFactor in the list recommendation, but lags behind in the item recommendation. LIRE is able to utilize the relationship between a list and its contained items. The recommendation performance of item can strengthen that of list. This is why LIRE outperforms CoFactor in the list recommendation. However, the co-occurrence relationships among items are ignored in LIRE, but are incorporated in CoFactor. This is why CoFactor outperforms LIRE in the item recommendation. 3) The recommendation performance of BPR and CoFactor in the list recommendation are quite similar. User's consumption process over items always show sequential feature, but the sequential feature is not that much obvious in the list consumption process (a user always consumes the list individually). The co-occurrence relationships among lists is not that much significant. This is why CoFactor does not show great improvement over BPR.

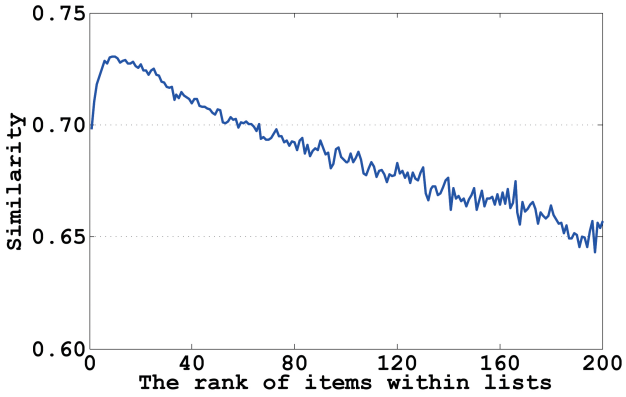


Figure 6: The similarity between the list and its contained items.

5.6 Importance of Items within A List (RQ5)

As discussed in word embedding algorithms [18, 25], the syntactic analogies and the semantic analogies of words and sentences can be found by computing the cosine distance among them. Once we have obtained the embedding representations of both items and lists, we can compute the similarity (the similarity is equal to 1 - distance) between a list and its contained words. In our datasets, we only know the rank of each item within a list, but we are not sure whether the items within a list are equally important. To answer this question, we explored the similarity between the embedding representation of a list and the embedding representations of its contained items. If the similarity between a list and an item is relatively high, it shows the item is relatively representative and important for the list.

In our datasets, the minimum and maximum length of lists is 10 and 996, respectively. The average length of lists is 77.8. Figure 6 shows the average similarity results between lists and their top 200 ranked items. As can be seen, the similarity increases first and then decreases, and reaches its maximum when the rank of item is around 10. This finding is relatively novel. First of all, the most representative item within a list is not ranked in the top of the list but in around the 10th position. It is mainly because when a user browses a list, his/her attention concentrates on items around the 10th position and then decides whether to favour the list. Meanwhile, the similarity between the list and its contained items gradually decreases when the rank of item increases beyond 10. This is mainly because the user's attention gradually faded with the increasing rank of items within the list.

6 CONCLUSION AND FUTURE WORK

This paper presents novel embedding factorization models for jointly recommending user generated lists and their contained items. Our methods nicely combine factorization models and embedding-based algorithms. Particularly, EFM-Side employs the list as side-information to discover the co-occurrence relationships among items. As a byproduct, it is capable of solving the new-item cold-start problem, where items are not yet consumed by users but exist in lists. By utilizing user interactions with item

and list simultaneously, the recommendation performance of user-item and user-list can be mutually reinforced under the EFM-Joint framework. To validate the effectiveness of our proposed approaches, we constructed two benchmark datasets. Experiment results over these two datasets have demonstrated the effectiveness of our work. We also performed micro-analysis to show whether the items within a list are equally important, and whether EFM-Joint is able to find the most representative item in a list.

In future, we plan to extend our work in the following three directions: 1) Modeling the sequential feature in user generated sequential behaviors. Although the sequential behavior is ignored in user generated lists, sequential feature is extremely important in some real-world scenarios (*e.g.,* music listening behavior and visiting tourist attractions). 2) Realizing the item and list recommendation in an online setting [14, 39]. Users' personal interests evolve over time, so do the content of user generated lists and their contained items. As it is computationally prohibitive to retrain a recommender model online, it would be helpful to utilize users' reviews to capture the dynamic changes in an online learning manner. 3) Modeling multi-modal data. The current social web has been overwhelmed with user-generated images and videos. With recent advances on image understanding [34, 36], it is interesting to develop semantic list recommendation methods for multi-media objects.

7 ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (No. 61672442 and No. 61373147), and the Science and Technology Planning Project of Fujian Province (No. 2016Y0079). This work is also supported by NExT research centre, which is supported by the National Research Foundation, Prime Ministers office, Singapore under its IRC@SG Funding Initiative.

REFERENCES

[1] Immanuel Bayer, Xiangnan He, Bhargav Kanagal, and Steffen Rendle. 2017. A Generic Coordinate Descent Framework for Learning from Implicit Feedback. In *WWW*. ACM.
[2] Da Cao, Xiangnan He, Liqiang Nie, Xiaochi Wei, Xia Hu, Shunxiang Wu, and Tat-Seng Chua. 2017. Cross-Platform App Recommendation by Jointly Modeling Ratings and Texts. *ACM Transactions on Information Systems* (2017).
[3] Da Cao, Liqiang Nie, Xiangnan He, Xiaochi Wei, Jialie Shen, Shunxiang Wu, and Tat-Seng Chua. 2016. Version-sensitive mobile App recommendation. *Information Science* 381 (2016), 161–175.
[4] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 2017. Attentive Collaborative Filtering: Multimedia Recommendation with Feature- and Item-level Attention. In *SIGIR*. ACM.
[5] Tao Chen, Xiangnan He, and Min-Yen Kan. 2016. Context-aware Image Tweet Modelling and Recommendation. In *MM*. ACM, 1018–1027.
[6] Zhiyong Cheng and Jialie Shen. 2016. On effective location-aware music recommendation. *ACM Transactions on Information Systems* 34, 2 (2016), 13.
[7] Zhiyong Cheng, Jialie Shen, and Steven CH Hoi. 2016. On Effective Personalized Music Retrieval by Exploring Online User Behaviors. In *SIGIR*. ACM, 125–134.
[8] Nemanja Djuric, Hao Wu, Vladan Radosavljevic, Mihajlo Grbovic, and Narayan Bhamidipati. 2015. Hierarchical neural language models for joint representation of streaming documents and their content. In *WWW*. ACM, 248–255.
[9] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, Steffen Rendle, and Lars Schmidt-Thieme. 2010. Learning attribute-to-feature mappings for cold-start recommendations. In *ICDM*. IEEE, 176–185.
[10] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. 2015. E-commerce in Your Inbox: Product Recommendations at Scale. In *SIGKDD*. ACM, 1809–1818.
[11] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Predictive Analytics. In *SIGIR*. ACM.
[12] Xiangnan He, Ming Gao, Min-Yen Kan, and Dingxian Wang. 2017. BiRank: Towards Ranking on Bipartite Graphs. *IEEE Transactions on Knowledge and Data Engineering* 29, 1 (2017), 57–71.
[13] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *WWW*. ACM.
[14] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *SIGIR*. ACM, 549–558.
[15] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *SIGKDD*. ACM, 426–434.
[16] Yehuda Koren, Robert Bell, Chris Volinsky, and others. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.
[17] Artus Krohn-Grimberghe, Lucas Drumond, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2012. Multi-relational matrix factorization using bayesian personalized ranking for social network data. In *WSDM*. ACM, 173–182.
[18] Quoc V Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *ICML*. ACM, 1188–1196.
[19] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *NIPS*.
[20] Dawen Liang, Jaan Altosaar, Laurent Charlin, and David M Blei. 2016. Factorization Meets the Item Embedding: Regularizing Matrix Factorization with Item Co-occurrence. In *RecSys*. ACM, 59–66.
[21] Jovian Lin, Kazunari Sugiyama, Min-Yen Kan, and Tat-Seng Chua. 2013. Addressing cold-start in app recommendation: latent user models constructed from twitter followers. In *SIGIR*. ACM, 283–292.
[22] Jovian Lin, Kazunari Sugiyama, Min-Yen Kan, and Tat-Seng Chua. 2014. New and improved: modeling versions to improve app recommendation. In *SIGIR*. ACM, 647–656.
[23] Xin Liu, Yong Liu, and Xiaoli Li. 2016. Exploring the Context of Locations for Personalized Location Recommendations. In *IJCAI*. AAAI Press, 1188–1194.
[24] Yidan Liu, Min Xie, and Laks VS Lakshmanan. 2014. Recommending user generated item lists. In *RecSys*. ACM, 185–192.
[25] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*. MIT, 3111–3119.
[26] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*. AUAI, 452–461.
[27] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. Fast context-aware recommendations with factorization machines. In *SIGIR*. ACM, 635–644.
[28] Oren Sar Shalom, Noam Koenigstein, Ulrich Paquet, and Hastagiri P Vanchinathan. 2016. Beyond Collaborative Filtering: The List Recommendation Problem. In *WWW*. ACM, 63–72.
[29] Chaofeng Sha, Xiaowei Wu, and Junyu Niu. 2016. A Framework for Recommending Relevant and Diverse Items. In *IJCAI*. AAAI Press, 3868–3874.
[30] Fumin Shen, Wei Liu, Shaoting Zhang, Yang Yang, and Heng Tao Shen. 2015. Learning binary codes for maximum inner product search. In *ICCV*. IEEE, 4148–4156.
[31] Ajit P Singh and Geoffrey J Gordon. 2008. Relational learning via collective matrix factorization. In *SIGKDD*. ACM, 650–658.
[32] Flavian Vasile, Elena Smirnova, and Alexis Conneau. 2016. Meta-Prod2Vec: Product Embeddings Using Side-Information for Recommendation. In *RecSys*. ACM, 225–232.
[33] Meng Wang, Weijie Fu, Shijie Hao, Hengchang Liu, and Xindong Wu. 2017. Learning on Big Graph: Label Inference and Regularization with Anchor Hierarchy. *IEEE Transactions on Knowledge and Data Engineering* 29, 5 (2017), 1101–1114.
[34] Meng Wang, Xueliang Liu, and Xindong Wu. 2015. Visual Classification by l1-Hypergraph Modeling. *IEEE Transactions on Knowledge and Data Engineering* 27, 9 (2015), 2564–2574.
[35] Suhang Wang, Jiliang Tang, Yilin Wang, and Huan Liu. 2015. Exploring Implicit Hierarchical Structures for Recommender Systems. In *IJCAI*. AAAI Press, 1813–1819.
[36] Hanwang Zhang, Xindi Shang, Huanbo Luan, Meng Wang, and Tat-Seng Chua. 2016. Learning from collective intelligence: Feature learning using social images and tags. *ACM Transactions on Multimedia Computing, Communications, and Applications* 13, 1 (2016), 1.
[37] Hanwang Zhang, Fumin Shen, Wei Liu, Xiangnan He, Huanbo Luan, and Tat-Seng Chua. 2016. Discrete collaborative filtering. In *SIGIR*. ACM, 325–334.
[38] Pengfei Zhao and Dik Lun Lee. 2016. How Much Novelty is Relevant? It Depends on Your Curiosity. In *SIGIR*. ACM, 315–324.
[39] Zhou Zhao, Hanqing Lu, Deng Cai, Xiaofei He, and Yueting Zhuang. 2016. User Preference Learning for Online Social Recommendation. *IEEE Transactions on Knowledge and Data Engineering* 28, 9 (2016), 2522–2534.
[40] Zhou Zhao, Ruihua Song, Xing Xie, Xiaofei He, and Yueting Zhuang. 2015. Mobile Query Recommendation via Tensor Function Learning.. In *IJCAI*, Vol. 15. AAAI Press, 4084–4090.
[41] Tao Zhu, Patrick Harrington, Junjun Li, and Lei Tang. 2014. Bundle recommendation in ecommerce. In *SIGIR*. ACM, 657–666.