Information Sciences 411 (2017) 122–135

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

TIIREC: A tensor approach for tag-driven item recommendation with sparse user generated content

Lu Yu[a,b], Junming Huang[c], Ge Zhou[a], Chuang Liu[a], Zi-Ke Zhang[a,*]

[a] Alibaba Research Center for Complexity Sciences, Hangzhou Normal University, Hangzhou 311121, China
[b] Computer, Electrical and Mathematical Sciences & Engineering, King Abdullah University of Science and Technology, Saudi Arabia
[c] Web Sciences Center, University of Electronic Science and Technology of China, China

ARTICLE INFO

Article history:
Received 2 May 2016
Revised 10 May 2017
Accepted 16 May 2017
Available online 17 May 2017

Keywords:
Recommender systems
Collaborative filtering
Matrix factorization
Latent factor model
Collaborative retrieval
Top-k ranking

ABSTRACT

In recent years, tagging system has become a building block o summarize the content of items for further functions like retrieval or personalized recommendation in various web applications. One nontrivial requirement is to precisely deliver a list of suitable items when users interact with the systems via inputing a specific tag (i.e. a query term). Different from traditional recommender systems, we need deal with a *collaborative retrieval* (CR) problem, where both characteristics of retrieval and recommendation should be considered to model a ternary relationship involved with *query*× *user*× *item*. Recently, several works are proposed to study CR task from users' perspective. However, they miss a significant challenge raising from the sparse content of items. In this work, we argue that items will suffer from the sparsity problem more severely than users, since items are usually observed with fewer features to support a feature-based or content-based algorithm. To tackle this problem, we aim to sufficiently explore the sophisticated relationship of each *query*× *user*× *item* triple from items' perspective. By integrating item-based collaborative information for this joint task, we present an alternative factorized model that could better evaluate the ranks of those items with sparse information for the given query–user pair. In addition, we suggest to employ a recently proposed *Bayesian Personalized Ranking* (BPR) algorithm to optimize *latent collaborative retrieval* problem from pairwise learning perspective. The experimental results on two real-world datasets, (i.e. *Last.fm, Yelp*), verified the efficiency and effectiveness of our proposed approach at top-k ranking metric.

1. Introduction

In recent years, massive applications like Last.fm, LinkedIn, Douban.com build tagging system to offer alternative approach for users to expose the features of items. For example, users in LinkedIn can recommend tags to show someone's proficient professional skills, and Last.fm users can allocate some keywords to highlight the musical style of artists. The emergence of tags makes content providers conveniently organise the web stuff, meanwhile allows users to search some interested information based on a given tag. For example, Last.fm as an example, users may hope to seek a cluster of interested rock artists via querying the "rock" tag, and douban users may expect to find some action movies based on the "Action Movie" tag to spend a nice weekend. Different from traditional user–item recommendation task, such requirement

* Corresponding authors.
E-mail addresses: liuchuang@hznu.edu.cn (C. Liu), zhangzike@gmail.com (Z.-K. Zhang).

http://dx.doi.org/10.1016/j.ins.2017.05.025
0020-0255/

involves with a joint problem of producing recommendations to a particular user with a specific tag (or query). This kind of task could be regarded as tag-driven item recommendation, which is actually an instance of *collaborative retrieval (CR)* [37]. In this situation, the tag can be referred as the users' input query since the target indexing term is the selected tag by users.

In order to deal with collaborative retrieval problem, approaches should be capable of modeling various patterns from the *query*× *user*× *item* triplets instead of the traditional *user*× *item* matrix. Usually, collaborative retrieval system suffers from more severe data sparsity than traditional tag-free recommendation services. In addition to insufficient users' click behaviors over items, another serious challenge is the lack of content features to present the aspects of items, or even worse without being attached with related tags. However, such phenomenon seem to frequently appear in real application. If we miss the consideration of this situation, many items could be unfairly evaluated in a ranked list due to the missing tags to show its relationship with the input querying tag by users. By now, several pioneering works are capable to model triple relationship like *query*× *user*× *item*. For examples, tensor factorization models [7,27,33,35] are proposed to extend collaborative filtering to recommend items in a tensor setup, while usually designed for special recommendation tasks like tag, mobile app, etc. Among different tensor models, Tucker Decomposition (TD) [35] encodes ternary relation into cubic factorization dimension. The drawback of using TD is that only cubic interaction of feature vectors is modeled while pairwise matching problem also exists in *query*× *user*× *item* triples to emphasize the contribution of a pair entities to the whole triple interactions. As a special case of TD, canonical decomposition (CD) [7] also has the same problem as TD. Rendle et al. [29] proposed that the pairwise interactions should be explicitly modeled to solve ternary tensor recommendation problem. Hariri et al. [11] extended the Latent Dirichlet Allocation (LDA) to jointly model the users, items, and the meta-data at topic level, while ignoring a fact that items could seriously lack of content feature to represent its characteristics.

Recently, some works are proposed to implement *collaborative retrieval* task based on matrix factorization techniques. They represent *query*× *user*× *item* as a user-central tripartite in hoping to leverage user-based collaborative network to represent the ternary relationship of *query*× *user*× *item*. Jason et al. [37] early explored the application of tensor models, and proposed the *Latent Collaborative Retrieval* (LCR) method. This method leverages user–user similarity to relief the pain of sparsity problem caused by the sparse online users click data. However, they discard a significant challenge, i.e. the *asymmetric information* of item side. In this work, we argue that (1) items suffer from the sparsity problem more severely than users, since items are usually observed with fewer features to support a feature-based or content-based algorithm; (2) users are dynamic while items are relatively static, which makes user–user similarity less stable and reliable than item–item similarity.

Items are not independent, which is indicated by an early work in [19]. Connection among them could be explored via users' collaborative behaviors, through which similar items will have more common users' to show interests on them. In addition, the characteristics of a single entity can not only be expressed by its own feature vector, but also implicitly represented by its neighbor connections from heterogeneous entities. For instance, an item's latent feature can be denoted as a combination of its own tag content represented as a low-dimension feature vector, where each dimension reveals the significance of a latent factor. Inspired by this intuition, we propose a general approach called *ternary interactive item recommendation* (TIIREC) via leveraging item–item connections to better express items' intrinsic feature. In particular, we also explore the power of those implicit connections involved with *user–query–item*. In the proposed method, a user's feature vector is a linear combination of feature vectors of those entities which have direct connections observed from the dataset. Viewing items as media vertices, we utilize item's neighbors with similar tastes to overcome the sparsity problem caused by the lack of content features.

The TIIREC model is trained with a recent proposed pairwise learning algorithm *Bayesian Personalized Ranking* (BPR) [28], which has been applied in many published recommendation tasks including tag recommendation [29], focused matrix factorization for advertisement [16]. As pointed out in [13,36] and verified on a real dataset *Last.fm*[1] [6], BPR significantly reduces the runtime required to train a latent model while keeping the same performance, compared with the training strategy *Weighted Approximate-Rank Pairwise* (WARP) [37] that was previously used in LCR.

In summary, the main contributions of this paper include:

- Comparing with WARP learning algorithm, the application of BPR algorithm on optimizing the parameters of LCR can sharply increase the training speed and simultaneously preserve almost the same performance on evaluation metric.
- We consider the item-based collaborative information to sufficiently alleviate the sparsity problem caused by items' lack of content features, and subsequently propose the TIIREC method.
- The experimental results on the *Last.fm* and *Yelp* datasets show that the proposed algorithm TIIREC is superior to baseline algorithms, especially, when dealing with a dataset containing a massive amount of items with sparse information.

The remainder of this paper is organized as follows. Section 2 describes the related works. Preliminaries are introduced at Section 3. In Section 4 we detail research motivation and the proposed recommendation model. Experimental results are given in Section 5. Finally, Section 6 summarizes this work and outlooks future work.

[1] http://www.last.fm.

Table 1
Symbols used in this paper.

Symbol	Definition and description
$\mathcal{Q}$	Set of queries
$\mathcal{A}$	Set of items
$\mathcal{U}$	Set of users
$\mathbf{X}$	Training set
D_X	Pairwise training samples
$\sigma(\cdot)$	Sigmoid function
$f(q, u, a)$	Scoring function for measuring user u's preference on item a with respect to a given query q
S	Query feature matrix
T	Item feature matrix
V	User feature matrix
U	User encoder matrix
A	Item encoder matrix
n	Feature dimension parameter
Θ	Parameter space

2. Related work

Recommendation and retrieval techniques have become essential components of massive applications, like e-commerce, search engine, location-based social network etc. In information retrieval, expected permutation of items or documents usually depend on the correlation between the content features of items and a given query by users. Many retrieval algorithms are proposed to capture semantic relevance of query-document, like Latent Semantic Indexing [8], LDA [5] topic model for low-dimensional representation of the word. In addition, factorized models like Polynomial Semantic Indexing (PSI) [2] are also employed to implement the task of document retrieval. However, they're usually towards optimizing the AUC ranking loss, but not top-k ranked list. In recommendation field, many works based on factorized models are proposed to produce predictions to active users. In particular, Matrix Factorization-based methods [15,18,32,40] are very close to our method since each entity (e.g. user, item, query) is represented as a low-dimensional feature vector. He et al. [12] review matrix factorization problem from element-wise perspective, and proposed a fast bath optimization approach for point wise ranking problem. Zhang et al. [42] presented an interesting work on extracting user collaborative network to make factorization more efficient on dealing with different recommendation problems. They are all general framework, but only dealing with bipartite networks. While in tag-driven tasks, search procedure begins with a tag, then items will be recommended according to users' historical behaviors over both tags and items. Traditional methods working on bipartite network can be applied on user–item or item–tag, but not directly capturing triplet relationship. Several types of approaches are available to model triple relationship, such as classical Tucker decomposition [35], PARAFAC [10], compact latent factor model [22], translated-based embedding [4]. Many context-aware collaborative filtering techniques are also proposed for such multi-relationship learning task, in particular contextual information related to users, like tags [29], web pages [24], demographics [17], temporal effects [38], multimedia retrieval for image innovation [25], information diffusioin [43] etc. Rendle et al. [28] turned to pairwise ranking optimization other than only rating estimation for recommendation problems with only implicit feedback. It is a seminal research on bringing rating estimation to pairwise learning in recommendation field. Lin et al. [21] presented theoretical analysis on different pairwise loss function, and proposed a new objective function for pairwise learning. According to the authors in [30,41] dynamic pairwise sampling strategy has been proposed to further improve Bayesian Personalized Ranking framework for top-k recommendation performance. In addition, recent works on heterogeneous information network [23,26,31,39,44] can also be adaptive to give vision into the sophisticated interactions hidden in the complicated dataset (Table 1).

3. Preliminaries

In this section, problem definition will firstly be given to briefly describe the concepts of collaborative retrieval, then a classical type of latent factor model will be described to present how to capture triple relationship from tensor perspective.

3.1. Problem definition

The objective of *collaborative retrieval* can be simply defined as generating a personalized ranking list of items to fit a particular user's tastes with respect to a given query. To achieve this goal, the proposed approaches should clearly define a scoring function $f(\cdot)$ to represent the relevance of a given triple (query,user,item) $\in \mathcal{Q} \times \mathcal{U} \times \mathcal{A}$, where $\mathcal{Q}$, $\mathcal{U}$, $\mathcal{A}$ denote the set of queries, users, items, respectively. In practice, only the top-k retrieved items could draw users' attention. Thereby, the learned scoring function $f(\cdot)$ should promote users' interesting items to high position as much as possible for a particular query. Generally, the parameters of $f(\cdot)$ can be derived from the training samples by optimizing a pre-defined ranking loss function.

3.2. Latent collaborative retrieval

The central idea of latent collaborative retrieval (LCR) is to represent each entity (e.g. user, query, item) as a n-dimensional feature vector. Analogous to matrix factorization approaches [18], the relationship between each pair entities can be measured by the dot-product of their latent factor vector. Formally, LCR's parameter space includes matrices $S \in \mathbb{R}^{|\mathcal{Q}|\times n}$, $V \in \mathbb{R}^{|\mathcal{U}|\times n}$, and $T \in \mathbb{R}^{|\mathcal{A}|\times n}$, which denotes the feature matrix of queries, users, and items, respectively. To preciously evaluate a user's preference on a item with respect to a given query, LCR additionally allocates each user u an encoder matrix $U_u \in \mathbb{R}^{n\times n}$. The scoring function $f(\cdot)$ of LCR model can be then given as follows:

$$f(q,u,a) = S_q U_u T_a^\top + V_u T_a^\top, \tag{1}$$

where S_q represents the row of S corresponding to query q, V_u is the row of V corresponding to user u, and T_a denotes the row of T corresponding to item a. Intuitively, the first term in Eq. (1) distinguishes LCR from the proposed tensor models for context-aware CF[2] [1,3,17,29] tasks. The second term independent of the query denotes users' basic preferences over items.

4. Ternary interactive item recommendation

In this section, we will firstly describe our motivation behind the proposed method. Then we will show how to measure the relevance of *(user, query, item)* triple after graphically revisiting the relationship of *(user, query, item)*.

4.1. Motivation

In real applications, we often suffer from the problem of *asymmetric information*, that is to say, characteristics of a majority of items can not be fully expressed due to the lack of specific description. By contrast, massive amount of popular items are represented as adequate descriptive terms. In this circumstance, the retrieval systems would unfairly evaluate the ranks of those items with sparse information since current description can not cover the characteristics of them. In order to cope with this problem, many web applications, such as *delicious*,[3] *douban*,[4] integrate *collaborative tagging* function [14], which offers privileges for users to add free-style tags to the shared content, e.g. books, movies, videos. To a certain extent, collaborative tagging enriches the content of items with descriptive terms for future filtering or search. However, there are still a huge amount of items with sparse information because of the lack of motivated users to share their tags. Therefore, the dataset that we have involved mainly contains items either represented as adequate tags, or oppositely with few ones.

To promote the performance of retrieval systems, we assume that the characteristics of those items with sparse information, denoted as *sparse items*, can be derived from their neighbors with abundant descriptive terms. Typically, one can firstly employ efficient tag recommendation approaches [9,14] to pre-target sparse items with most possible keywords or tags. Then, retrieval systems could return a ranked list of items by computing the scores for all items with respect to a given user–query pair. However, the pre-targeting process could cost lots of computation resources. This significant challenge impulses us to think "Can we invent some algorithms to naturally represent the ternary interaction of (*query, user, item*) in collaborative retrieval task?"

To answer this question, we next graphically review the relationship of (*user, query, item*) and interpret how to model the ternary relationship from both users' and items' perspectives.

4.2. Revisit ternary relationship

To formulate our ideal, we firstly represent the relationship of the *(user, query, item)* triple as a graph, simply shown in Fig. 1(a), where entities (*user, query, item*) are represented as vertices and red edges represent the observed relationship between a pair of entities. In order to intuitively describe the ternary relationship of (*user, query, item*), we decompose Fig. 1(a) into two parts showed in Fig. 1(b) and Fig. 1(c), respectively. In Fig. 1(b), each positive observation $(user, query, item) \in \mathbf{X}$ can be represented as a route path $query - user - item$,[5] where users serve as the intermedia vertices responsible to transform the "resource" between items and queries, and the bipartite graph in the right intuitively represents the users' preferences over items. In terms of Fig. 1(c), the relationship of each positive observation is represented as a route path $user - item - query$, where differently items serve as the intermedia "resource" transformer between users and queries, and the left bipartite graph represent the rating information of user–item pairs. In the following part, we will give insight into the possible effects brought by the slight difference between Fig. 1(b) and (c).

It can be seen that Fig. 1(b) illustrates an user-central behavior network, where the relationship of user–item pairs is represented as a collaborative network in the right part, analogously, the left part depicts the collaborative interactions of user–query pairs. In practical application, Fig. 1(b) could be extremely sparse due to the tinily available querying or

[2] The goal of context-aware CF is to produce personalized recommendations to a particular user by taking into account the contextual information, such as time, location, and so on. Therefore, the query could also be regarded as a context factor.

[3] https://delicious.com/.

[4] http://www.douban.com.

[5] Here we use "path" to distinguish the representation of (*user, query, item*) relationship from different perspectives.

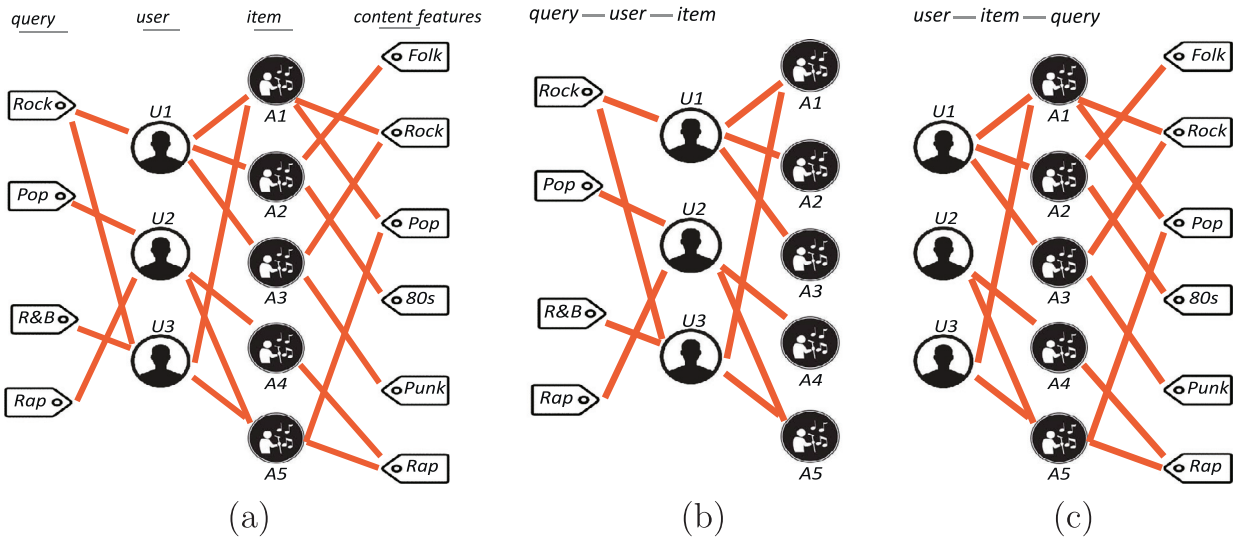

Fig. 1. Graphical representation of the *user* × *query*× *item* relationship.

rating information from users. In order to deal with such challenge caused by the lack of users' historical behaviors, the proposed approach for collaborative retrieval task should combine both collaborative networks in Fig. 1(b). By doing this, users' preferences over unconnected queries or items could be induced from their neighbors that have similar querying bias, or tend to rate a similar set of items.

Turning to Fig. 1(c), we can easily find that left part is shared with Fig. 1(b), however it represents a collaborative network from items' perspective. Analogous to Fig. 1(b), the right part of Fig. 1(c) represents the collaborative network of item–query pairs. Since content features of items might be the possible queries, the word "query" in Fig. 1(c) generally indicate both content features and the observed queries. With the problem of *asymmetric information* mentioned in Section 3.1, massive amount of items in Fig. 1(c) actually have too limited content to represent intrinsic feature. To better measure the ternary relationship of (*user, query, item*), this challenge should be taken into account when we attempt to design an innovative algorithm for collaborative retrieval task. Going through basic idea of recommendation problem, we find that many benefits of item-based methods have been fully discussed. However, it is still lack of attention on utilization of collaborative network from item side to cope with sparsity problem for collaborative retrieval task. We propose that leveraging neighbors with similar features to induce the target item's unknown features might be the possible way to help us to better evaluate the rank of sparse items with respect to a given user–query pair. However, the case in collaborative retrieval task is totally different from traditional recommendation problem, which usually involves binary relationship between user–item pairs, rather than the ternary interactions among (*user, query, item*). As we mentioned in Section 3.1, pre-targeting items could cost a lot of computation in searching similar items. In this work, we prefer to combine both user–item and item–query collaborative networks to better capture the latent relationship of *user* – *item* – *query*. We believe that it could further tackle the sparsity problem caused by the lack of content features, meanwhile improve the effect of learning the ternary relevance of *user* – *item* – *query* path.

4.3. Our method

As mentioned above, we consider items' collaborative information based on the assumption that regarding items as the media vertex could leverage the similar items [34] with rich descriptive terms to improve the performance of retrieval systems on estimating the ranks of sparse items. In our model, the score of *user* – *item* – *query* path in Fig. 1(c) can be given by:

$$g(q, a, u) = S_q A_a V_u^\top, \tag{2}$$

where $A_a \in \mathbb{R}^{n\times n}$ is the linear transformation matrix of item a. Consequently, we integrate the item-central triple relevance into the Eq. (1), then TIIREC can be modified as:

$$f(q, u, a) = S_q U_u T_a^\top + V_u T_a^\top + S_q A_a V_u^\top \tag{3}$$

where the first term intuitively captures the ternary interaction of (*query, user, item*) by using encoder matrix U_u of each user to indirectly combine both user–query and user–item collaborative networks into one model. The querying and rating preferences of target users could be learned based on not only their own historical behaviors, but also their neighbors'. To some extent, it could effectively deal with the sparsity problem caused by the lack of users' historical information. Differently, the third term in Eq. (3) focuses on modeling items' collaborative information to tackle the sparsity problem caused

by the items' content features. As we allocate an encoder matrix with dimension $n \times n$ to each user and item, one might argue that the complexity of TIIREC could be extremely huge. To give a response the concern, we define a global user-encoder matrix U instead of modifying an encoder matrix U_u to every single user. The reason behind this is that we think sparsity problems raised from item-side could have more impacts on building a precise predictor than the user-side. By doing this, we only need to maintain a $U \in \mathbb{R}^{n\times n}$ for the user-side collaborative network, and the item encoders $A \in \mathbb{R}^{|\mathcal{T}|\times n\times n}$ can ensure the power of TIIREC to capture the patterns hide in the *(user, query, item)* observations. Then the modified relevance function f of TIIREC can be presented as:

$$f(q,u,a) = S_q U T_a^\top + V_u T_a^\top + S_q A_a V_u^\top \tag{4}$$

In addition, we could see that each user has connected to a group of queries and items. To some extent, a part of user's interest could be represented by the group of historical input queries, and the feature of the clicked items can also be another intermediate to reveal partial characteristics of users. Based on the assumption, we further modify both items' and users' low-dimensional vectors as follows:

$$\begin{aligned}\tilde{V}_u &= V_u + \frac{1}{\sqrt{|\mathcal{Q}_u|}}\sum_{q\in\mathcal{Q}_u} S_q + \frac{1}{\sqrt{|\mathcal{A}_u|}}\sum_{a\in\mathcal{A}_u} T_a \\ \tilde{T}_a &= T_a + \frac{1}{\sqrt{|Q_a|}}\sum_{q\in\mathcal{Q}_a} S_q.\end{aligned} \tag{5}$$

where $\mathcal{Q}_u$ and $\mathcal{A}_u$ denote a set of queries and items that are ever input by user u, respectively. $\mathcal{Q}_a$ denotes a set of observed queries correlated to item a. With the modified feature vectors, the final relevance of function for TIIREC can be defined as follows:

$$f(q,u,a) = S_q U \tilde{T}_a^\top + \tilde{V}_u \tilde{T}_a^\top + S_q A_a \tilde{V}_u^\top \tag{6}$$

In the next section we are going to describe a pairwise learning algorithm to induce our model from a ranking perspective. The experimental results in Section 4.4 show that TIIREC can better estimate the ranks of items with respect to a given user–query pair after considering the item-based collaborative information. The possible reason might be that TIIREC explicitly models item-based interactions.

4.4. Pairwise learning approaches

The unknown parameters of the scoring function should be efficiently learned under the assumption that the top-k retrieved items should include as many profitable selections as possible for a particular user with respect to a given query. To achieve this, the CR task can be generally formulated as generating a ranked list of items for a given (u, q) pair by solving a pairwise ranking problem. A common approach is to regard each observation in a given training set $\mathbf{X}$, including m observations $(q_i, u_i, a_i)_{i=1,2,\ldots,m} \in \mathcal{Q}\times\mathcal{U}\times\mathcal{A}$, as a positive retrieval event, otherwise as a negative example. Subsequently, the selected learning algorithm should give the definition of *pairwise violation* cost, which would be produced if a negative item is assigned with a larger score or within a "margin" from the positive item.

In this section, we shall firstly detail the mechanism of WARP, originally used in the first piece of CR work [37], then introduce a generic learning algorithm proposed by Rendle et al. [28], namely BPR. Finally, we will show how to adapt BPR instead of WARP to effectively learn the relevance function of LCR and the proposed method, TIIREC.

4.4.1. Weighted Approximate-Ranking Pairwise (WARP)

The WARP loss can be defined as follows:

$$err_{WARP} = \sum_{i=1}^{m} L(rank_{a_i}(\bar{f}(q_i,u_i))), \tag{7}$$

where $\bar{f}(q_i,u_i)$ is a vector, which contains predictions for all items for a fixed user–query pair. The a_ith element of $\bar{f}(q_i,u_i)$, denoted as $\bar{f}_{a_i}(q_i,u_i)$, is the predicted relevance value of ith training example (q_i, u_i, a_i). Correspondingly, $rank_{a_i}(\bar{f}(q_i,u_i))$ in Eq. (7) is a margin-based rank of item a_i, defined as:

$$rank_{a_i}(\bar{f}(q_i,u_i)) = \sum_{b\neq a_i} \mathbb{I}[1 + \bar{f}_b(q_i,u_i) \geq \bar{f}_{a_i}(q_i,u_i)], \tag{8}$$

where $\mathbb{I}[\cdot]$ is the indicator function. In Eq. (7), L is a weight function evaluating the loss of the current scoring function f:

$$L(k) = \sum_{i=1}^{k} \alpha_i, \ \ with\ \alpha_1 \geq \alpha_2 \geq \alpha_3 \geq \cdots \geq 0. \tag{9}$$

According to the suggestion of Weston et al. [37], we choose $\alpha_i = 1/i$ as the weighting approach, which would assign large weights to top positions with rapidly decaying weight for lower positions. Intuitively, optimizing the WARP loss means to

rank each positive item a_i in the training set to highest position. For example, given a random query–user pair denoted as (q_i, u_i), if the score of an uncollected item b is less than a margin of one from the score of a_i, this pair will produce a cost.

Subsequently, Eq. (7) could be optimized by gradient descent based algorithms. However, in each updating step, it is expensive to compute the exact value of $rank_{a_i}$ for each observation when the number of items is very large. Thus, the exact rank of Eq. (8) can be estimated by a random sampling process at each step [37]. That is to say, for a given observed sample (q_i, u_i, a_i), one uniformly draws at random items from $\mathcal{A}$ until finding a violated item b, which satisfies $1 + f(q_i, u_i, b) > f(q_i, u_i, a_i)$, and then the rank of a_i can be approximated as

$$rank_{a_i}(\bar{f}(q_i, u_i)) \approx \lfloor \frac{|\mathcal{A}| - 1}{K} \rfloor, \tag{10}$$

where $\lfloor \cdot \rfloor$ is the floor function and K is the number of steps needed to find a item b. Then Eq. (7) can be modified as

$$\begin{aligned} \overline{err_{WARP}} &= \sum_{i=1}^{m} L_i \\ L_i &= L(rank_{a_i}(\bar{f}(q_i, u_i))) \cdot |1 - f(q_i, u_i, a_i) + f(q_i, u_i, b)|. \end{aligned} \tag{11}$$

Finally, Eq. (11) could be optimized by stochastic gradient descent (SGD). We constrain the parameters using $||S_i|| \le C$, $i \in \{1, \ldots, |\mathcal{Q}|\}$, $||V_i|| \le C$, $i \in \{1, \ldots, |\mathcal{U}|\}$, $||T_i|| \le C$, $i \in \{1, \ldots, |\mathcal{A}|\}$ and project the parameters back into the constraints at each SGD step.

4.4.2. Bayesian Personalized Ranking (BPR)

To clearly describe BPR algorithm [28], we leverage a notation D_X to denote the pairwise ranking constraints.

$$D_X = \{(u, q, a, b) : (u, q, a) \in \mathbf{X} \wedge (u, q, b) \notin \mathbf{X}\}$$

Next, we present a generic approach to solve the personalized ranking problem for CR tasks by maximizing the following posterior probability of the parameter space Θ.

$$p(\Theta| >_{u,q}) \propto p(>_{u,q} |\Theta) p(\Theta), \tag{12}$$

where $p(\Theta)$ denotes the prior probability of Θ, and notation $>_{u,q} = \{a >_{u,q} b : (u, q, a) \in \mathbf{X}, (u, q, b) \notin \mathbf{X}\}$ denotes the pairwise ranking structure for a given (u, q) pair. We assume that each element of $>_{u,q}$ is independently drawn from the same probability. Hence, the above likelihood function $p(>_{u,q}|\Theta)$ can be rewritten as:

$$p(>_{u,q} |\Theta) = \prod_{(u,q,a,b) \in D_X} p(a >_{u,q} b|\Theta), \tag{13}$$

where $p(a >_{u,q} b|\Theta)$ denotes the probability that a user really prefers item a to item b for a given query, defined as [28]:

$$\begin{aligned} p(a >_{u,q} b|\Theta) &= \sigma(\hat{x}_{u,q,a,b}(\Theta)) \\ \sigma(x) &= \frac{1}{1 + e^{-x}}. \end{aligned} \tag{14}$$

Here we choose $\hat{x}_{u,q,a,b}(\Theta) = \hat{x}_{u,q,a} - \hat{x}_{u,q,b}$, in fact $\hat{x}_{u,q,a,b}(\Theta)$ could be a arbitrary real-valued function depending on the parameters Θ. For $p(\Theta)$, we define it as a normal distribution with zero mean and covariance matrix $\sum_\Theta = \lambda_\Theta I$, that is, $\Theta \backsim \mathcal{N}(\mathbf{0}, \sum_\Theta)$. Now we can infer the BPR-Opt by filling $p(\Theta)$ into the maximum posterior probability in Eq. (12).

$$\begin{aligned} BPR\text{-}\textsc{Opt} &= \ln p(\Theta| >_{u,q}) \\ &= \ln \prod_{(u,q,a,b) \in D_X} p(a >_{u,q} b|\Theta) p(\Theta) \\ &= \ln \prod_{(u,q,a,b) \in D_X} \sigma(\hat{x}_{u,q,a,b}(\Theta)) p(\Theta) \\ &= \sum_{(u,q,a,b) \in D_X} \ln \sigma(\hat{x}_{u,q,a,b}) + \ln p(\Theta) \\ &= \sum_{(u,q,a,b) \in D_X} \ln \sigma(\hat{x}_{u,q,a,b}) - \lambda_\Theta ||\Theta||^2, \end{aligned}$$

where λ_Θ are model regularization parameters.

Typically, gradient ascent based algorithm is an apparent optimization strategy for maximizing the posterior probability in Eq. (12). However, standard gradient ascent is not suitable for BPR-Opt because we have to compute $\hat{x}_{u,q,a,b}$ for all negative items b with respect to a given training sample $(u, q, a) \in \mathbf{X}$. If we have a large amount of items, it will be inefficient to update the parameters in each gradient ascent step. To effectively learn the parameters, a stochastic gradient-ascent (SGA) based algorithm, namely LearnBPR , was proposed by Rendle et al. [28] for optimizing BPR-Opt. Instead of comparing with all negative items, LearnBPR only draws at random a negative item b in each SGA step, which makes *BPR* superior to *WARP* in terms of the computation complexity (see next section). The procedure of *BPR* is presented in Algorithm 1.

Algorithm 1 Optimizing models for BPR with LearnBPR.

Input: **X**: Training Set
Output: Learned $\hat{\Theta}$
1: initialize Θ
2: **repeat**
3: randomly draw (u,q,a) from **X**
4: randomly draw (u,q,b) from $(u \times q \times \mathcal{A}) \setminus \mathbf{X}$
5: $\hat{x}_{u,q,a,b} \leftarrow \hat{x}_{u,q,a} - \hat{x}_{u,q,b}$
6: $\Theta \leftarrow \Theta + \alpha((1 - \sigma(\hat{x}_{u,q,a,b}))\frac{\partial}{\partial\Theta}\hat{x}_{u,q,a,b} - \lambda_\Theta\Theta)$
7: **until** convergence.

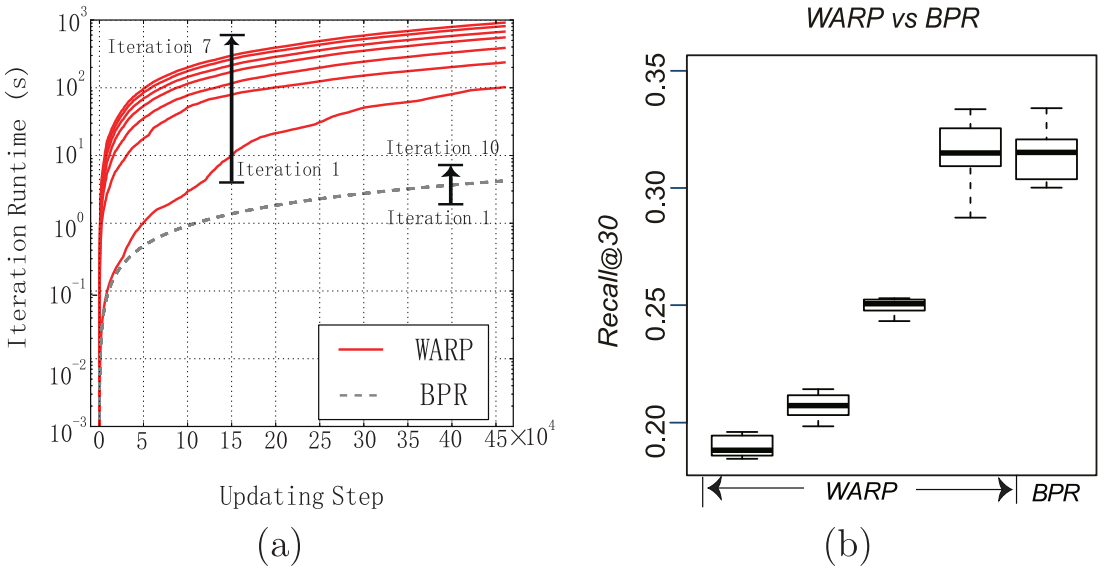

Fig. 2. (a) Accumulated runtime required by WARP and BPR for searching violated items in each training iteration. (b) Comparisons of WARP and BPR on recall@30. Four groups of (Learning rate α, constraint C) setups are randomly selected for WARP. One group of (Learning rate α, regularization λ) setup chosen for BPR mimics the best performance of WARP. The performance of each learning algorithm is validated on the same testing dataset.

4.5. Learning LCR and TIIREC with BPR

In above sections, we intuitively review two optional learning algorithms. In this section, we will present the comparative results to study the efficiency of both learning algorithms.

In terms of WARP, the random sampling procedure dominates the runtime according to several recent works [13,36]. Weston et al. [36] indicated that each SGD step requires less than $1 + \min(\frac{|\mathcal{A}|-1}{rank_a(f(u,q))})$ sampling times to find a violated item b on average. However, the computations of scores on items b could increase sharply or even worse for a massive item database, if the positive item a happens to rank at the top of the list. Interestingly, such occasional situation seems to happen frequently according to Hsiao et al. [13]. Comparatively, BPR only samples one time at each parameter updating step. To demonstrate the efficiency of BPR for optimizing the LCR model, we conduct several experiments with the same evaluation metric *Recall@30* on the dataset *Last.fm* described in Section 4.1. To avoid the fluctuations coming from parameters randomly initialization on the results, each experiment for Fig. 2(b) runs ten independent times. Experimental results show that BPR needs less runtime to learn the parameters of LCR than WARP (see Fig. 2(a)) without decreasing the performance of LCR on the evaluation metric (see Fig. 2(b)). In Fig. 2(a), the iteration runtime curves of BPR are difficult to be distinguished from each other, thus runtime curves of ten training iterations are crowded together.

In summary, we propose to employ BPR instead of WARP to optimize LCR model. The complete BPR learning procedure for LCR can be found in Algorithm 2, where the function of 5–12th steps is to update corresponding parameters based on the 5–6th steps in Algorithm 1.

As Eq. (6) shows, the model parameters of TIIREC include:

$$T \in \mathbb{R}^{|\mathcal{A}|\times n}, V \in \mathbb{R}^{|\mathcal{V}|\times n}, S \in \mathbb{R}^{|\mathcal{Q}|\times n}$$
$$A \in \mathbb{R}^{|\mathcal{A}|\times n\times n}, U \in \mathbb{R}^{n\times n}$$

Algorithm 2 Optimizing LCR with BPR.

Input: **X**: Training Set
Output: S, U, V, T

Initialize S, U, V, T from uniform probability $\mathcal{U}(x, y)$
repeat
  for (u,q,a) in **X do**
    randomly draw (u,q,b) from $(u \times q \times \mathcal{A}) \setminus \mathbf{X}$
    $\hat{x}_{u,q,a,b} \leftarrow f(u, q, a) - f(u, q, b)$ with Eq. (1)
    Updating S_q, U_u, V_u, T_a, T_b
    $loss \leftarrow 1 - \sigma(\hat{x}_{u,q,a,b})$
    $S_q \leftarrow S_q + \alpha(loss \cdot U_u(T_a - T_b)^\top - \lambda S_q)$
    $V_u \leftarrow V_u + \alpha(loss \cdot (T_a - T_b) - \lambda V_u)$
    $T_a \leftarrow T_a + \alpha(loss \cdot (S_q U_u + V_u) - \lambda T_a)$
    $T_b \leftarrow T_b - \alpha(loss \cdot (S_q U_u + V_u) + \lambda T_b)$
    $U_u \leftarrow U_u + \alpha(loss \cdot S_q^\top (T_a - T_b) - \lambda U_u)$
  end for
until validation performance does not improve.

For a given training dataset, the values of parameters in TIIREC can be learned by following learning procedure of BPR. As we can see, we need to calculate the gradients for each involved parameters under the BPR framework. The complete learning procedure of TIIREC is described in Algorithm 3, where the function of 5–14th steps is to update corresponding

Algorithm 3 Optimizing TIIREC with BPR.

Input: **X**: Training Set
Output: S, V, U, T, A

Initialize S, V, U, T, A from uniform probability $\mathcal{U}(x, y)$
repeat
  for (u, q, a) in **X do**
    randomly draw (u,q,b) from $(u \times q \times \mathcal{A}) \setminus \mathbf{X}$
    $\hat{x}_{u,q,a,b} \leftarrow f(u, q, a) - f(u, q, b)$ with Eq. (6)
    Updating $S_q, V_u, T_a, T_b, U_u, A_a, A_b$
    $S_q \leftarrow S_q + \alpha((1 - \sigma(\hat{x}_{u,q,a,b})) \cdot (U(\tilde{T}_a - \tilde{T}_b)^\top + (A_a - A_b)\tilde{V}_u^\top) - \lambda S_q)$
    $V_u \leftarrow V_u + \alpha((1 - \sigma(\hat{x}_{u,q,a,b})) \cdot (\tilde{T}_a - \tilde{T}_b + S_q(A_a - A_b)) - \lambda V_u)$
    $T_a \leftarrow T_a + \alpha((1 - \sigma(\hat{x}_{u,q,a,b})) \cdot (S_q U + \tilde{V}_u) - \lambda T_a)$
    $T_b \leftarrow T_b - \alpha((1 - \sigma(\hat{x}_{u,q,a,b})) \cdot (S_q U + \tilde{V}_u) + \lambda T_b)$
    $U \leftarrow U + \alpha((1 - \sigma(\hat{x}_{u,q,a,b})) \cdot S_q^\top (\tilde{T}_a - \tilde{T}_b) - \lambda U)$
    $A_a \leftarrow A_a + \alpha((1 - \sigma(\hat{x}_{u,q,a,b})) \cdot S_q^\top \tilde{V}_u - \lambda A_a)$
    $A_b \leftarrow A_b - \alpha((1 - \sigma(\hat{x}_{u,q,a,b})) \cdot S_q^\top \tilde{V}_u + \lambda A_b)$
  end for
until validation performance does not improve.

parameters based on the 5–6th steps in Algorithm 1. Complete implementation of TIIREC can be found at https://github.com/jeppe/TIIREC.

5. Experiments

In this section, we present experimental results to demonstrate the efficiency of the proposed collaborative retrieval model (TIIREC) on two real-world datasets, one of which is obtained from the Last.fm music website, the other one is from *Yelp Dataset Challenge*[6] Round 3.

5.1. Dataset and preprocessing

To implement CR task, we need to preprocess the dataset. It is noted that the query analysis applications will transform the provided queries as keywords or phrases which are the basic units to represent the items. Then retrieval system return a ranked list of items based on the parsed queries. Thereby, a common approach is to regard a *query*× *user*× *item* triple as

[6] http://www.yelp.com/dataset_challenge.

Table 2
Basic statistics of *Lastfm* and *Yelp* dataset.

	lastfm-50tags	Yelp
#Users	1529	16,826
#Item	8669	14,902
#Query	50	587
#Observed feedback	574,521	806,261
#Density	8.67×10^{-4}	5.47×10^{-6}

Table 3
Parameters used for each method.

Method	$last.fm - 50tags$	*Yelp*
TIIREC	α=0.04, λ=0.01	α=0.08, λ=0.01
LCR	α=0.04, λ=0.01	α=0.1, λ=0.01
PITF	α=0.002, λ=0.01	α=0.02, λ=0.01

a *keyword*$\times$ *user*$\times$ *item*, where the keywords are equivalent to the set of filtered user tags, or content features. We carry out data preprocessing to obtain two expectant datasets for the comparison experiments.

Last.fm: This dataset was released in the framework of HetRec 2011, called $hetrec2011 - last.fm - 2k$ ($last.fm - 2k$) [6], and contains heterogeneous information, which mainly covers users' collaborative tagging behaviors, listening preferences on artists, as well as social relationship.

In this paper, we focus on resources associated with collaborative retrieval tasks, that is, users' listening and tagging logs, which contain more than ten thousand unique tags utilized by 1892 users to reveal the characteristics of 17,632 listened artists. As we need to validate the ability of the proposed methods on retrieving sparse items, we then construct a comparatively denser dataset from Lastfm. More specially, we keep only observations related to the top 50 most used tags, generally correlated to the genres of music: for example, the top 5 tags are "rock", "pop", "alternative", "electronic", and "indie".

If an artist a has ever been listened and assigned with several tags by a user u, for example, *rock* and *indie*, then we allocate ($rock, u, a$) and ($indie, u, a$) to $last.fm - 50tags$. If the artist is not assigned with any tags by the user u, the genres, assigned by other users to a, are distributed to the (u, a) pairs. If no user has ever assigned any genre to a, we choose to drop corresponding (u, a) pairs in default, since such user–artist pairs are not perfect training examples for the CR task. In the end, we infer the $last.fm - 50tags$ (see Table 2) with 574,521 data points of the form (u, q, a) from the $last.fm - 2k$ dataset.

Yelp: In addition to $last.fm - 2k$ dataset, this research is also performed on a recent academic dataset, published under the licence of *Yelp Dataset Challenge* Round 3, hereafter referred to as *Yelp*. This dataset contains 335,022 reviews and ratings given by 70,746 users to 15,470 businesses located in the Phoenix and AZ metropolitan area. Each business is characterized with rich content, for example, category like "Restaurants", "Shopping", "Health & Medical", which makes us accessible to have insight into users' preferences over different businesses. We pre-filter this data to contain users with at least 4 reviews, also corresponding businesses. After pre-filtering, we obtain a processed dataset including 806,261 well-formed points with format (user, category, business), given by 16,826 users to 14,902 businesses, which totally are pre-tagged with 587 business categories. For example, top-10 categories are "Restaurants", "Shopping", "Food", "Beauty & Spas", "Automotive", "Mexican", "Health & Medical", "Home Services", "Nightlife", "Fashion". Each sample (user, category, business) indicates that a particular user ever rated a business associated with the target category.

5.2. Experimental setup

In this paper, we conduct numerous experiments to evaluate the performances of each algorithm on the real Last.fm and Yelp datasets. In terms of Last.fm dataset, we randomly draw 80% of samples in $last.fm - 50tags$ for training, 10% for validation, and the rest for testing. Based on this dataset, we study the efficiency of WARP and BPR on optimizing LCR model (see Section 3.4). The performance is evaluated on the testing dataset. Analogously, we randomly draw 60% of samples in *Yelp* for training, 20% for validation, and the rest for testing. The initial values of feature matrices of both models are randomly drawn from a uniform distribution $\mathcal{U}(-0.02, 0.02)$. The hyperparameters α and λ are chosen for each algorithm using the validation set respectively. Different experiment settings are used based on the dataset and algorithms (see Table 3).

Our experiments are conducted to address the following issues:

1. How does our proposed method compare with relevant personalized ranking approaches?
2. Can the proposed algorithm improve the performance on evaluating the ranks of those items with few content features?

Table 4
Different algorithms' *Recall* vs. different k. The integer numbers of PITF and NMF represent the dimension n, and n is set as 10 for TIIREC and LCR algorithms.

Dataset	Method	Recall@5	Recall@10	Recall@15	Recall@20	Recall@25	Recall@30
Last.fm	NMF-30	0.0495	0.078	0.103	0.126	0.144	0.164
	PITF-10	0.073	0.12	0.163	0.195	0.223	0.255
	PITF-100	0.0878	0.137	0.175	0.217	0.242	0.275
	LCR	0.091	0.159	0.23	0.298	0.343	0.378
	TIIREC	**0.1**	**0.174**	**0.259**	**0.315**	**0.374**	**0.392**
Yelp	NMF-30	0.0133	0.0226	0.0248	0.0276	0.0356	0.0435
	LCR	0.031	0.042	0.0698	0.089	0.122	0.154
	PITF-10	0.033	0.073	0.104	0.186	0.216	0.246
	PITF-100	0.096	0.155	0.204	0.237	0.259	0.305
	TIIREC	**0.148**	**0.21**	**0.274**	**0.287**	**0.356**	**0.416**

Comparison methods: To demonstrate the efficiency of our proposed approach, we mainly compare TIIREC with the state-of-the-art collaborative retrieval algorithm, that is, LCR model as well as algorithms for tensor environmental and traditional collaborative filtering. Besides LCR, the following baseline algorithms are used in this paper:

- *Pairwise interaction tensor factorization (PITF)* [29]: As the state-of-the-art tensor algorithm for personalized tag recommendation, PITF is a personalized and context-aware algorithm aiming to return a top-k ranked list of tags when given a particular user–item pair. In PITF, the interaction between each entity pair is expressed as the dot-product of specific feature vectors. The ternary relationship *user*$\times$ *item* $\times$ *tag* is modified by the following score function:

$$\hat{x}(u,t,i) = \hat{u}_u^T \hat{t}_t^U + \hat{u}_u^I \hat{i}_i^U + \hat{t}_t^I \hat{i}_i^T, \tag{15}$$

  where the first term $\hat{u}_u^T \hat{t}_t^U$ denotes the relevance value of the given user u and tag t, middle term $\hat{u}_u^I \hat{i}_i^U$ denotes the relevance value of the given user u and item i, and the last term $\hat{t}_t^I \hat{i}_i^T$ denotes the interaction between the given item i and tag t. According to Rendle and Schmidt-Thieme [29], the user–item interaction term vanishes when LearnBPR is employed to optimize top-k ranking task. Thus, the final score function for tag recommendation task is:

$$\hat{x}(u,t,i) = \hat{u}_u \hat{t}_t^U + \hat{i}_i \hat{t}_t^I. \tag{16}$$

  Analogously, collaborative retrieval task involves with returning a top-k ranked list of items with respect to a given user–query pair, which inspires us to adapt it to CR tasks by lightly modifying Eq. (15). Likewise, the user–query interaction term vanishes, then the modified score function is denoted as:

$$\hat{x}(u,q,i) = \hat{u}_u \hat{i}_i^U + \hat{q}_q \hat{i}_i^Q. \tag{17}$$

  where the first term $\hat{u}_u \hat{i}_i^U$ denotes the interaction between user u and item i, and the last term denotes the interaction between query q and item i. The values of parameters can be leaned by employing BPR algorithm [29].
- *Non-negative Matrix Factorization (NMF)* [20]: This approach is not directly applied to model ternary relationship. In this paper, we perform NMF on the item-query matrix to compute a top-k ranked preference items for given q and u. The NMF implementation we used in this paper is from http://www.csie.ntu.edu.tw/~cjlin/nmf/.
- *LCR* [37]: This method is the first piece of work for CR task. The authors try to measure the triple relevance of (*query, user, item*) via the integration of user-central collaborative network.

Evaluation metric: According to Ref. [37], the performance of each algorithm is measured by recall@k, a widely used metric to evaluate the recommendation accuracy in top-k. For a given testing example (q, u, a), we first compute $f(q, u, i)$ for each item $i \in \mathcal{A}$, and then sort them in descending order of score. Then recall@k equals to 1 if item a appears in the top-k list, and equals to 0 otherwise. We report mean recall@k over the whole test dataset.

$$Recall@K = \frac{\sum_{u,q \in \mathcal{U} \times \mathcal{Q}} \#of\ recalled\ items}{|N|} \tag{18}$$

5.3. Performance evaluation

The comparison experiments are conducted on two dataset with different density scale. Table 2 summarizes the basic information of the observed datasets. It is noted that the $last fm - 50 tags$ is approximately over 158 times denser than *Yelp* dataset, which offers available stuff to distinguish the performance of our proposed model from other baseline algorithms in dealing with sparsity problem. We mainly consider the performance of algorithms on different settings of the values of k and feature dimension n. In addition, we also validate the effectiveness and efficiency of LCR and TIIREC on evaluating the ranks of sparse items in top-k list.

Validation results: Table 4 presents the comparison results of all approaches on both Last.fm and Yelp datasets with different length of recommendation list. Here we fix the number of latent factors of LCR and TIIREC to 10. From the results,

Table 5
Recall@30 vs. the dimension parameter n.

Dataset	Method	$n = 10$	$n = 50$	$n = 100$	$n = 200$
Last.fm	LCR	0.378	0.49	0.502	0.518
	TIIREC	**0.392**	**0.505**	**0.525**	**0.532**
Yelp	LCR	0.154	0.334	0.419	0.459
	TIIREC	**0.416**	**0.497**	**0.534**	**0.571**

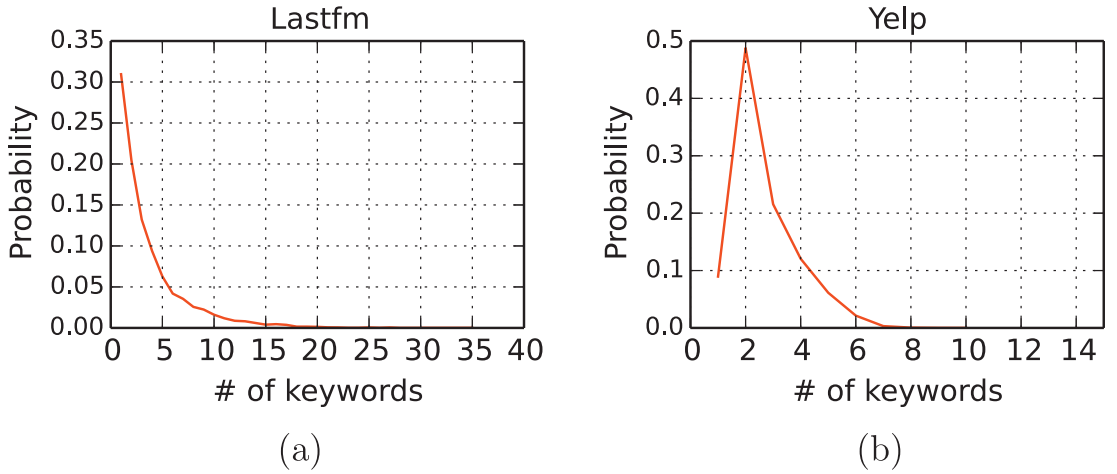

Fig. 3. Probability distribution of training dataset according to the number of keywords that an item contains.

we can see that the correctly predicted items of all algorithms increase as the growth of k. Comparing with other baseline algorithms, TIIREC performs best in retrieving items with given queries by users. One possible reason might be that TIIREC explicitly models user- and item-central collaborative network, which makes TIIREC sufficiently estimate the relevance of (*user, query, item*). Although LCR models the ternary relationship of (*user, query, item*), we argue that it fails to leverage the item-based tripartite (see Fig. 1(c)) to effectively measure users' preference over items with given queries. In terms of Last.fm dataset, TIIREC increases at least 3.7% when k=30 in comparison with LCR, whilst improves over 12% in the best case when k equals to 15. Especially in Yelp dataset which has more sparse items, the performance of LCR significantly decreases with different settings. As for PITF, it proposes to explicitly use pairwise interactions to represent the ternary relationship. However, interaction between a user and a query is insufficiently learned due to the vanishment of user–query term under the BPR learning framework. NMF performs worst on both datasets, which indicates that only modeling item–query bipartite network is not enough to deal with the CR task, that is, how to effectively measure the relevance of a triple (*user, query, item*). Table 5 shows that the performance of both TIIREC and LCR improves along with the increase of dimension n. Comparing with Last.fm, the performance gap between TIIREC and LCR is quite evident in *Yelp* dataset. It indirectly offers some evidences that the integration of item-based information might be useful when the dataset includes massive amount of sparse items.

Sparse items ranking performance. In Section 3, we argue that the asymmetric information problem can have items with less keywords ranked unfairly due to the lack of content to express their characteristics. In this partition, we perform experiments to investigate whether the proposed method can improve the ranking performance for sparse items with few observable content features in the datasets. Fig. 3 shows the probability distribution of selecting an item according to the number of keywords. According to Fig. 3, we can see that different datasets exhibit different distribution. In Last.fm dataset, although some items have more than 10 keywords, over 80% of items have fewer than 5 keywords. Therefore, we modify those items with less than 5 keywords as sparse items for Last.fm dataset. In Yelp dataset, items with two keywords is quite large, almost takes up 50%, and over 80% of items have fewer than 4 keywords. In this case, we denote items with less than 4 keywords as the sparse items for Yelp dataset. It's noted that only 10% of testing cases are sparse items in Last.fm, in contrast with over 70% of testing cases are sparse items in Yelp. Table 6 shows the validation results of all algorithms on recalling sparse items in top-k recommendation list. We can see that our proposed method significantly outperforms other methods on both datasets. Moreover, when comparing the performance in Yelp, the improvement of TIIREC indicates that integration of item-based collaborative information is more suitable for sparse collaborative retrieval task.

Table 6
Sparse items ranking performance at Recall@K.

Dataset	Method	*Recall*@20	*Recall*@30
Last.fm	NMF	0.00018	0.00378
	PITF-10	0.00059	0.00129
	PITF-100	0.00018	0.00036
	LCR	0.0093	0.0293
	TIIREC	**0.0191**	**0.0477**
Yelp	NMF	0.0279	0.0365
	LCR	0.08	0.147
	PITF-10	0.171	0.221
	PITF-100	0.207	0.276
	TIIREC	**0.254**	**0.372**

6. Conclusions and future works

We aim to design an effective collaborative retrieval algorithm to objectively predict the ranks of those items with lack of descriptive terms to reveal their own basic characteristics. To achieve this goal, we focus on the profits of leveraging the collaborative information of items to better evaluate the ranks of those sparse items, and propose to express the latent sophisticated relationships for CR task from not only users' perspective, but also items' perspective. Then, we propose a superior latent collaborative retrieval model, TIIREC, by integrating the possible item-based information into LCR model. Finally, we study the primary principles of LCR, of which we find that the runtime of LCR is dominated by the chosen learning approach, namely WARP. Therefore, we propose to employ a generic approach, namely BPR, instead of WARP to optimize the LCR model on the basis of further experimental analysis covering the topics of training efficiency and accuracy.

The proposed model can be easily generalized to deal with many other tasks involving to model ternary interaction among entities such as collaborative image annotation, personalized search. Most typical example might be the tagging recommendation system, which aims to recommend tags to users with respect to items. As the emergence of knowledge graph, entity relationship is represented as a triplet which is ideally matching the proposed model. In future, we could explore the possibility of representing entity and relations from tensor perspective. However, we just explore possible interactions among (*user, query, item*) triple. In practice, relationships of a pair of entities always involve with massive ingredients, which could be termed as a currently-prevalent word, heterogeneous relationships. Due to this significant characteristics, we will attempt to make our proposed model adaptive to learn the multi-relation by exploring diverse aspects of a sophisticated system in future work.

Acknowledgments

This work was partially supported by National Natural Science Foundation of China (Grant nos. 61673151 and 61503110), Zhejiang Provincial Natural Science Foundation of China (Grant nos. LY14A050001 and LQ16F030006).

References

[1] G. Adomavicius, A. Tuzhilin, Context-aware recommender systems, context-aware recommender systems, Recommender Systems Handbook, Springer, 2011, pp. 217–253.
[2] B. Bai, J. Weston, D. Grangier, R. Collobert, K. Sadamasa, Y. Qi, C. Cortes, M. Mohri, Polynomial semantic indexing, in: Proceedings of Advances in Neural Information Processing Systems, 2009, pp. 64–72.
[3] L. Baltrunas, B. Ludwig, F. Ricci, Matrix factorization techniques for context aware recommendation, in: Proceedings of the ACM Conference on Recommender Systems, 2011, pp. 301–304.
[4] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: Proceedings of Advances in Neural Information Processing Systems, 2013, pp. 2787–2795.
[5] D.M. Blei, A.Y. Ng, M.I. Jordan, J. Lafferty, Latent dirichlet allocation, J. Mach. Learn. Res. 3 (2003) 993–1022.
[6] I. Cantador, P. Brusilovsky, T. Kuflik, Second workshop on information heterogeneity and fusion in recommender systems (hetrec2011), Proceedings of ACM Conference on Recommender Systems, Recsys (2011) 387–388.
[7] J.D. Carroll, J.-J. Chang, Analysis of Individual Differences in Multidimensional Scaling via an N-way Generalization of "Eckart-Young" Decomposition 35 3(1970) 283–319.
[8] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, R. Harshman, Indexing by latent semantic analysis, J. Am. Soc. Inf. Sci. 41 (1990) 391–407.
[9] N. Garg, I. Weber, Personalized, interactive tag recommendation for flickr, in: Proceedings of the ACM Conference on Recommender Systems, 2008, pp. 67–74.
[10] R.A. Harshman, Foundations of the Parafac procedure: Models and Conditions for an "Explanatory" Multimodal Factor Analysis (1970).
[11] N. Hariri, B. Mobasher, R. Burke, Query-driven context aware recommendation, in: Proceedings of the ACM Conference on Recommender Systems, 2013, pp. 9–16.
[12] X. He, H. Zhang, M.-Y. Kan, T.-S. Chua, Fast matrix factorization for online recommendation with implicit feedback, in: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2016, pp. 549–558.
[13] K.-J. Hsiao, A. Kulesza, A. Hero, Social collaborative retrieval, in: Proceedings of the 7th International Conference on Web Search and Data Mining, 2014, pp. 293–302.
[14] R. Jäschke, L. Marinho, A. Hotho, L. Schmidt-Thieme, G. Stumme, Tag recommendations in folksonomies, in: Proceedings of European Conference on Principles of Data Mining and Knowledge Discovery, 2007, pp. 506–514.
[15] Y. Koren, Factorization meets the neighborhood: a multifaceted collaborative filtering model, in: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2008, pp. 426–434.

[16] B. Kanagal, A. Ahmed, S. Pandey, V. Josifovski, L. Garcia-Pueyo, J. Yuan, Focused matrix factorization for audience selection in display advertising, in: Proceedings of the IEEE International Conference on Data Engineering, 2013, pp. 386–397.
[17] A. Karatzoglou, X. Amatriain, L. Baltrunas, N. Oliver, Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering, in: Proceedings of the ACM Conference on Recommender Systems, 2010, pp. 79–86.
[18] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (8) (2009) 30–37.
[19] G. Linden, B. Smith, J. York, Amazon.com recommendations: item-to-item collaborative filtering, IEEE Internet Comput. 7 (1) (2003) 76–80.
[20] D.D. Lee, H.S. Seung, Learning the parts of objects by non-negative matrix factorization, Nature 401 (6755) (1999) 788–791.
[21] J. Lin, Y. Lei, B. Zhang, D.-X. Zhou, Online pairwise learning algorithms with convex loss functions, Inf. Sci. 406–407 (2017) 57–70.
[22] C.-L. Liu, X.-W. Wu, Large-scale recommender system with compact latent factor model, Expert Syst. Appl. 64 (2016) 467–475.
[23] X. Luo, Y. Xia, Q. Zhu, Applying the learning rate adaptation to the matrix factorization based collaborative filtering, Knowl. Based Syst. 37 (2013) 154–164.
[24] A.K. Menon, K.P. Chitrapura, S. Garg, D. Agarwal, N. Kota, Response prediction using collaborative filtering with hierarchies and side-information, in: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2011, pp. 141–149.
[25] H. Ma, J. Zhu, M.R. Lyu, I. King, Bridging the semantic gap between image contents and tags, IEEE Trans. Multimed. 12 (5) (2010) 462–473.
[26] W. Pan, S. Xia, Z. Liu, X. Peng, Z. Ming, Mixed factorization for collaborative recommendation with heterogeneous explicit feedbacks, Inf. Sci. 332 (2016) 84–93.
[27] S. Rendle, L.B. Marinho, Alexandros nanopoulos, lars schmidt-thieme, learning optimal ranking with tensor factorization for tag recommendation, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2009, pp. 727–736.
[28] S. Rendle, C. Freudenthaler, Z. Gantner, Lars schmidt-thieme, BPR: Bayesian personalized ranking from implicit feedback, in: Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence, 2009, pp. 452–461.
[29] S. Rendle, L. Schmidt-Thieme, Pairwise interaction tensor factorization for personalized tag recommendation, in: Proceedings of the 3rd International Conference on Web search and Data Mining, 2010, pp. 81–90.
[30] S. Rendle, C. Freudenthaler, Improving pairwise learning for item recommendation from implicit feedback, in: Proceedings of the 7th International Conference on Web Search and Data Mining, 2014, pp. 273–282.
[31] Y. Sun, J. Han, X. Yan, P.S. Yu, T. Wu, Pathsim: meta path-based top-k similarity search in heterogeneous information networks, Proc. VLDB Endow. 4 (11) (2011) 992–1003.
[32] R. Salakhutdinov, A. Mnih, Probabilistic matrix factorization, in: Proceedings of Neural Information Processing Systems, 2007.
[33] P. Symeonidis, A. Nanopoulos, Y. Manolopoulos, Tag recommendations based on tensor dimensionality reduction, in: Proceedings of the Conference on Recommender systems, 2008, pp. 43–50.
[34] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of the 10th International Conference on World Wide Web, 2001, pp. 285–295.
[35] L.R. Tucker, Some mathematical notes on three-mode factor analysis, Psychometrika 31 (3) (1966) 279–311.
[36] J. Weston, S. Bengio, N. Usunier, Large scale image annotation: learning to rank with joint word-image embeddings, Mach. Learn. 81 (1) (2010) 21–35.
[37] J. Weston, C. Wang, R. Weiss, A. Berenzweig, Latent collaborative retrieval, in: Proceedings of the 29th International Conference on Machine Learning, 2012, pp. 9–16.
[38] L. Xiong, X. Chen, T.-K. Huang, J. Schneider, J.G. Carbonell, Temporal collaborative filtering with Bayesian probabilistic tensor factorization, in: Proceedings of SIAM International Conference on Data Mining, 2010, pp. 211–222.
[39] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, J. Han, Personalized entity recommendation: a heterogeneous information network approach, in: Proceedings of the 7th International Conference on Web Search and Data Mining, 2014, pp. 283–292.
[40] L. Yu, C. Liu, Z.-K. Zhang, Multi-linear interactive matrix factorization, Knowl. Based Syst. 85 (2015) 307–315.
[41] L. Yu, G. Zhou, C. Zhang, J. Huang, C. Liu, Z.-K. Zhang, RankMBPR: rank-aware mutual bayesian personalized ranking for item recommendation, in: Proceedings of 17th International Conference on Web-Age Information Management, 2016, pp. 244–256.
[42] C. Zhang, L. Yu, Y. Wang, C. Shah, X. Zhang, User collaborative network embedding for social recommender systems, in: Proceedings of the 17th SIAM International Conference on Data Mining, 2017.
[43] Z.-K. Zhang, C. Liu, X.-X. Zhan, X. Lu, C.-X. Zhang, Y.-C. Zhang, Dynamics of information diffusion and its applications on complex networks, Phys. Rep. 651 (2016) 1–34.
[44] B. Zou, C. Li, L. Tan, H. Chen, GPUTENSOR: efficient tensor factorization for context-aware recommendations, Inf. Sci. 299 (2015) 159–177.