Information Sciences 579 (2021) 55–71

Contents lists available at ScienceDirect

Information Sciences

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

Leveraging implicit relations for recommender systems

Anchen Li [a,b], Bo Yang [a,b,*], Huan Huo [c], Farookh Khadeer Hussain [c]

[a] College of Computer Science and Technology, Jilin University, China
[b] Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, China
[c] University of Technology Sydney, Australia

ARTICLE INFO

Article history:
Received 16 November 2020
Received in revised form 19 July 2021
Accepted 23 July 2021
Available online 27 July 2021

Keywords:
Recommender systems
Collaborative filtering
Implicit relations
Deep learning

ABSTRACT

Collaborative filtering (CF) is one of the dominant techniques used in recommender systems. Most CF-based methods treat every user (or item) as an isolated existence, without explicitly modeling potential mutual relations among users (or items), which are latent in user-item interactions. In this paper, we design a novel strategy to mine user-user and item-item implicit relations and propose a natural way of utilizing the implicit relations for recommendation. Specifically, our method contains two major phases: neighbor construction and recommendation framework. The first phase constructs an implicit neighbor set for each user and item according to historical user-item interaction. In the second phase, based on the constructed neighbor sets, we propose a deep framework to generate recommendations. We conduct extensive experiments with four datasets on the movie, business, book, and restaurant recommendations and compare our methods with seven baselines, e.g., feature-based, neighborhood-based, and graph-based models. The experiment results demonstrate that our method achieves superior performance in rating prediction and top-k recommendation.

1. Introduction

In the era of information explosion, recommender systems play an indispensable role in identifying user preferences by recommending products or services. Collaborative filtering (CF) is one of the state-of-art techniques in recommender systems [1–5]. In a typical CF scenario with user-item interaction history, matrix factorization (MF), which embeds users and items in a shared latent space and models the user preference to an item as the inner product between the corresponding user and item embeddings, has become one of the most popular approaches [6]. However, due to the complex interaction between users and items, the shallow representations in the MF-based methods lack the expressiveness to model features for users and items [7,8].

Recent years have witnessed the great success of deep neural network techniques in many research areas such as computer vision and natural language processing. Some recently proposed recommendation approaches utilize deep neural networks to capture the complex relationships between user-item interactions, which enhance the performance of the previous shallow models [7,9]. Though successful, most deep recommendation models treat every user (or item) as an isolated existence and have tended not to focus on potential user-user or item-item relations. Such potential relations are latent in user-item interactions and could provide valuable information to infer user or item features [8,10]. Although some existing works

* Corresponding author at: College of Computer Science and Technology, Jilin University, China.
E-mail address: ybo@jlu.edu.cn (B. Yang).

https://doi.org/10.1016/j.ins.2021.07.084
0020-0255/© 2021 Elsevier Inc. All rights reserved.

[11,12] utilize graph neural networks (GNNs) on the user-item bipartite graph to capture high-order relations among users (or items), a more explicit and straightforward way is to directly construct user-user and item-item relations. Empirical evidence is from the recent work MMCF [8], which explicitly utilizes the *co-occurrence* relation (i.e., users who have interacted with the same items or items with which the same users have interacted) to define the neighbors for users and items. For instance, Fig. 1(a) shows a simple user-item interaction in the movie domain, where each user rates movies on a 5-point integer scale to express their preference for movies. Take the user co-occurrence relation for example: for user *d*, the co-occurrence relation defines user *a*, user *b*, and user *c* as her neighbors (as shown in Fig. 1(b)) because user *d* and these users have interacted with common items.

Although MMCF has shown promising results, we argue that such a co-occurrence relation is macro-level and coarse-grained. For instance, for user *d*, the co-occurrence relation defines user *e* as one of her neighbors, but user *d* and user *e* have different preferences for movie *d*. By observing the user-item interaction in Fig. 1(a), although user *d* and user *a* are not in a co-occurrence relation, they both share common preferences with user *b* and user *c*. Such a high-order transitive relation is also a very significant signal for revealing user preferences and item properties, while it is ignored by most existing works. Therefore, we believe that (i) the co-occurrence relation contains useful information, but not all co-occurrence relations help; (ii) using the co-occurrence relation directly without filtering may introduce some irrelevant information or even noise, which will mislead the learning of user and item feature representations; (iii) the co-occurrence relation ignores high-order transitive relations for users and items.

To overcome the aforementioned limitation, we propose our method IRec which leverages user-user and item-item **I**mplicit relations for **Rec**ommendation. IRec contains two major phases: neighbor construction and a recommendation framework. In the first phase, we construct an implicit neighbor set for each user and each item. More specifically, we first utilize the user-item interaction information to construct a user relational graph and an item relational graph. We then map each graph to a latent continuous space to find the implicit neighbors for users and items. In this way, the constructed neighborhoods not only filter out some irrelevant (or noisy) co-occurrence relations, they also may contain high-order transitive relations. In the second phase, we design a deep framework based on graph neural networks (GNNs) which utilizes the constructed user and item neighbor sets for recommendation. The key component of the framework is that we devise an aggregator on the neighbor sets to update the feature representations of users and items. Empirically, we apply IRec to four real-world scenarios of the movie, business, book, and restaurant recommendations. The experiment results show that IRec outperforms the state-of-the-art approaches in both rating prediction and top-*k* recommendation. In summary, our main contributions in this paper are listed as follows:

- We provide a novel approach to find implicit neighbors for users and items.
- We propose an end-to-end framework that integrates implicit neighbors into recommendations.
- The experimental results on four real-world datasets show the effectiveness of IRec.

The remainder of this paper is organized as follows. Section 2 reviews the work related to our methods. In Section 3, we present the problem formulation and introduce the proposed method IRec. In Section 4, we describe the experiments conducted on four real-world datasets and present the experiment results, followed by a conclusion and suggestions for future work in Section 5.

2. Related work

In this section, we provide a brief overview of four areas that are highly relevant to our work.

2.1. Collaborative filtering

Collaborative filtering (CF) can generally be grouped into three categories: neighborhood-based model, latent factor model, and hybrid model [13]. Neighborhood-based methods identify neighborhoods of similar users or items based on the user-item interaction history [10,13]. For example, ItemKNN utilizes collaborative item-item similarities (e.g. cosine similarity) to generate recommendations [14]. The latent factor model, such as matrix factorization [6], projects users and items into low-dimensional feature vector spaces. The interactions between users and items are modelled as the inner product of their latent vectors. With the development of deep learning, some latent factor models utilize deep neural networks as representation learning tools to capture complex user-item interactions [7,9]. As for the hybrid model, it merges the latent factor model and the neighborhood-based model. SVD++ is a well-known and commonly used hybrid model that leverages users' explicit feedback and implicit feedback to predict user preferences [13]. Recently, a line of work leverages co-occurrence relations to define the neighbors for users and items and integrates deep components into the hybrid model [8,15]. Since the co-occurrence relation is coarse-grained and lacks high-order semantics, these methods are insufficient to generate better recommendations. Different from the aforementioned work, our proposed method IRec is a unified hybrid model using implicit relations to define user and item neighbors, which accounts for both co-occurrence relations and high-order transitive relations.

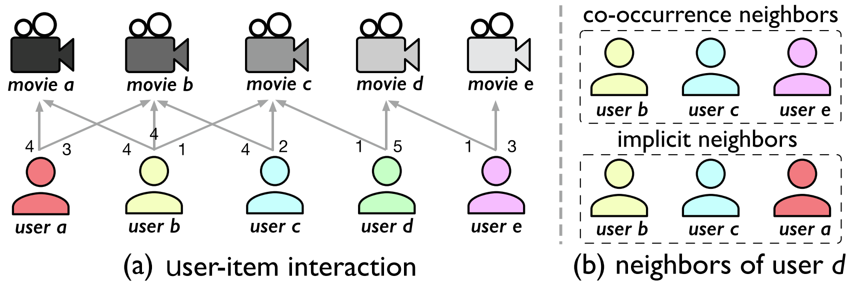


Fig. 1. Illustration of the problem. (a): A simple user-item interaction scenario in a 5-star system. (b): Comparison between the co-occurrence neighbors and implicit neighbors.

2.2. Graph representation

Graph representation learning is a significant method to learn latent, low-dimensional representations of vertexes in the graph, while preserving both graph structure and node content. In general, there are two types of graph representation learning methods: unsupervised methods and semi-supervised methods [16]. Unsupervised graph representation approaches focus on preserving the graph topology structure [17,18]. For instance, LINE designs the objective function that preserves first-order and second-order proximity for learning node representations [17]. DeepWalk utilizes local information obtained from random walks to learn node representations [18]. As for the semi-supervised method, it utilizes some labeled vertexes for representation learning [16]. Graph convolutional network (GCN) [19] and graph attention networks (GAT) [20] are two powerful semi-supervised methods for solving the classification problem in the graph. GCN learns node representation by aggregating the features of its neighbor nodes. GAT further enhances the performance of GCN by using the attention mechanism. In this work, we first use unsupervised representation approaches to construct the implicit neighbor sets. We then borrow the recent advances of GCN and design a framework that utilizes constructed neighbor sets for recommendation.

Some recent studies like GCMC [12], NGCF [11] also adopt GCN ideas for recommendation, and they are designed for the user-item bipartite graph. Different from the above literature, we provide a new perspective for recommendation with the assistance of constructed user-user and item-item implicit neighborhoods.

2.3. Deep learning

Deep learning is an emerging field of machine learning and is receiving a huge amount of attention at the moment [21]. Since deep learning technology can solve complex tasks while providing start-of-the-art results, it has achieved great success in many research areas [22], such as computer vision, speech recognition, and natural language processing. Due to the effectiveness of deep components, an increasing number of researchers are interested in integrating deep models into recommender systems [7–9,23–29]. For instance, NeuCF is proposed to model the user-item interactions with a multi-layer perceptron [7]. DKEN is a deep end-to-end framework that uses deep neural networks and knowledge graph embedding for knowledge-enhanced recommendation [24]. RM-DRL utilizes convolutional neural networks and recurrent neural networks to produce user and item semantic feature vectors, respectively [25]. R-ConvMF integrates convolutional neural networks into probabilistic matrix factorization for document-based recommendations [28]. LUAR designs a neural attention mechanism to find important auxiliary reviews to address the sparsity problem in review-based recommendation [29]. In this work, we propose a deep recommendation framework in our method IRec. The framework is based on graph neural networks. The key component of the framework is that we devise an aggregator on the neighbor sets to update the feature representations of users and items. With the help of deep learning, we can obtain sufficient representation power for building a successful recommender system.

2.4. Feedback information

The recommender systems collect user feedback information through the feedback techniques, and then utilize the feedback information to generate recommendations [30]. User feedback information can be roughly divided into two categories: explicit feedback (e.g., ratings) and implicit feedback (e.g., clicking and browsing history). There are several differences between the two types of feedback information [31]: (i) explicit feedback can capture both positive and negative user preferences, while implicit feedback can only be positive; (ii) compared with implicit feedback, explicit feedback can more accurately and unequivocally reflect users' interest in items; and (iii) explicit feedback is scarce and difficult to collect whereas implicit feedback is abundant and far outweighs the quantity of explicit data. To cater for different types of user feedback, researchers have designed corresponding explicit feedback recommendation methods [32–34] and implicit feedback recommendation methods [30,35]. Some studies also use both explicit and implicit data for personalized recommendations [9]. In this paper, we focus on explicit feedback recommender systems with users' rating information. We first utilize user explicit feedback to construct a neighbor set for each user and item. We then develop a deep framework that utilizes constructed neighbor sets for recommendation.

3. Methodology

In this section, we first introduce the notations and formulate the problems. We then describe two phases of IRec: neighbor construction and the recommendation framework.

3.1. Notations and problem formulation

In a typical recommendation scenario, we suppose there are M users $\mathcal{U} = \{u_1, u_2, \ldots, u_M\}$ and N items $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$. We define $\mathbf{Y} \in \mathbb{R}^{M \times N}$ as the user-item historical interaction matrix. If user u_a has rated item v_i before, the a, i-th element y_{ai} in $\mathbf{Y}$ is the rating score from u_a to v_i, otherwise we employ $y_{ai} = 0$ to represent the unknown rating.

Given the above information $(\mathcal{U}, \mathcal{V}, \mathbf{Y})$, the first phase of IRec outputs user relational data $\mathcal{N}_u = \{\mathcal{N}_u(1), \ldots, \mathcal{N}_u(M)\}$ and item relational data $\mathcal{N}_v = \{\mathcal{N}_v(1), \ldots, \mathcal{N}_v(N)\}$. $\mathcal{N}_u$ and $\mathcal{N}_v$ contain implicit neighbors for users and items respectively. The details of building up relational data $\mathcal{N}_u$ and $\mathcal{N}_v$ are discussed in subSection 3.2.

In the IRec's second phase, given the user-item interaction matrix $\mathbf{Y}$, user neighbor data $\mathcal{N}_u$ and item neighbor data $\mathcal{N}_v$, the recommendation framework aims to learn a prediction function $\hat{y}_{ai} = \mathcal{F}(u_a, v_i | \Theta, \mathbf{Y}, \mathcal{N}_u, \mathcal{N}_v)$, where $\hat{y}_{ai}$ is the predicted rating from user u_a to item v_i which she has never engaged before, and Θ is the framework parameters of function $\mathcal{F}$. The details of this phase are discussed in subSection 3.3.

3.2. Neighbor construction

In this subsection, we describe the neighbor construction strategy. This strategy contains three steps: (i) construct the user relational graph and item relational graph; (ii) map the user and item relational graphs to latent spaces respectively; and (iii) find implicit neighbors for users and items from their latent spaces. Next, we detail each step.

3.2.1. Step 1: construction of relational graphs

The user-item interaction data can be represented as a bipartite graph structure. We first transform the bipartite graph structure to construct a user relational graph and item relational graph to identify user-user relationships and item-item relationships. In addition, to reflect the strength of the relationship in a fine-grained way, we construct the relational graphs as weighted graphs. To this end, we utilize users' opinions on items to construct two weighted relational graphs $\mathcal{G}_u = (\mathcal{U}, \mathcal{E}_u)$ and $\mathcal{G}_v = (\mathcal{V}, \mathcal{E}_v)$ for users and items respectively. In the user relational graph $\mathcal{G}_u$, the edge $e^u \in \mathcal{E}_u$ connects two users if they have engaged with at least one common item before. In addition, e^u is associated with a weight $w^u > 0$ to indicate the relational strength between two users. Similarly, for the item relational graph $\mathcal{G}_v$, the edge $e^v \in \mathcal{E}_v$ connects two items if they have been engaged by at least one common user and its weight $w^v > 0$ indicates the relational strength between two items.

The opinions of users on items, such as ratings or reviews, play a crucial role in reflecting user preferences and item attributes. Here, we employ users' ratings to items as the user opinions. We utilize the difference in the user opinions to define the weight of the edges in the relational graphs. Specifically, for the edge e^u_{ab} between user u_a and user u_b in $\mathcal{G}_u$, the edge weight w^u_{ab} is defined as follows:

$$w^u_{ab} = Y_{max} - \frac{1}{|\mathcal{C}_{ab}|} \sum_{v_i \in \mathcal{C}_{ab}} |y_{ai} - y_{bi}|, \tag{1}$$

where Y_{max} is the max score in all ratings (e.g., 5 in a 5-star system) and $\mathcal{C}_{ab}$ is the subset of $\mathcal{V}$ containing the items that u_a and u_b both rated before. Similarly, the weight w^v_{ij} of the edge e^v_{ij} in $\mathcal{G}_v$ which connects item v_i and item v_j is defined as follows:

$$w^v_{ij} = Y_{max} - \frac{1}{|\mathcal{D}_{ij}|} \sum_{u_a \in \mathcal{D}_{ij}} |y_{ai} - y_{aj}|, \tag{2}$$

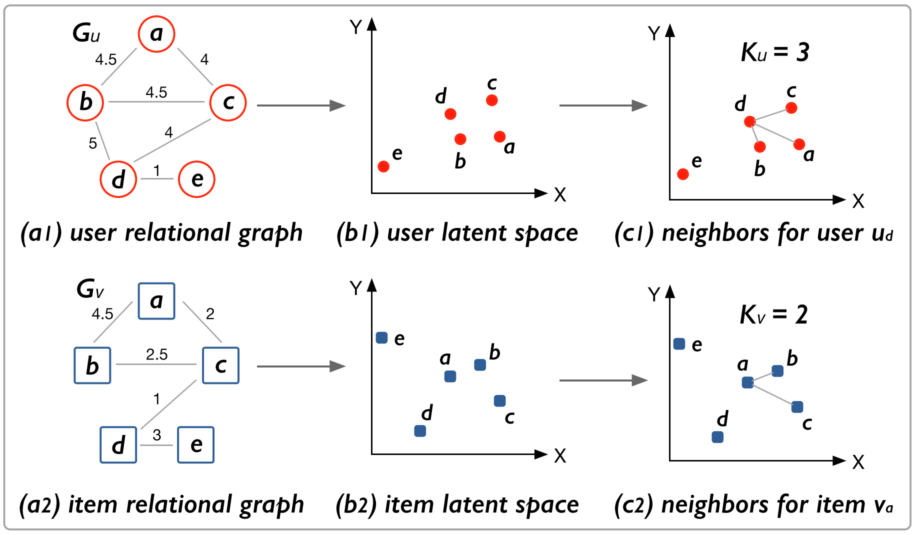
where $\mathcal{D}_{ij}$ is the subset of $\mathcal{U}$ containing the users who has rated both v_i and v_j before.

Fig. 2a1) and (a2) are toy examples of building relational graphs for users and items in the simple user-item interaction scenario (in Fig. 1 (a)). The advantage is that such relational graphs not only reflect the co-occurrence relations (one-hop neighbors) but also infer the high-order transitive relations (multi-hop neighbors).

For particular recommendation scenarios, one can employ other metrics to reflect users' opinions and design user-user and item-item opinion relations based on the metrics.

3.2.2. Step 2: relational graph mapping

After the relational graph construction, we utilize the node embedding method to map each relational graph to a latent continuous space. Specifically, for the user relational graph, we use a function $f_u : u \rightarrow z^u$ to map a user node $u \in \mathcal{U}$ from $\mathcal{G}_u$ to



Gu
a
b
c
d
e
4.5
4
4.5
5
4
1
Y
X
Ku = 3
(a1) user relational graph
(b1) user latent space
(c1) neighbors for user ud
Gv
4.5
2
2.5
1
3
Kv = 2
(a2) item relational graph
(b2) item latent space
(c2) neighbors for item va

Fig. 2. An illustration of the neighbor construction. (a) → (b): Mapping user and item relational graphs to latent spaces respectively. (b) → (c): Finding implicit neighbors for users and items respectively.

a low-dimensional vector $z^u \in \mathbb{R}^{l_u}$ in a latent continuous space, where l_u is the dimension number of the vector for users. Similarly, for the item relational graph, we utilize another function $f_v : v \to z^v$ to map an item node $v \in \mathcal{V}$ from $\mathcal{G}_v$ to a low-dimensional vector $z^v \in \mathbb{R}^{l_v}$, where l_v is the vector dimension for items. Note that z^u and z^v can also be considered as positions for user u and item v in their latent spaces. After the mapping, both the structures and properties of the relational graphs are preserved and presented as the geometry in the corresponding latent space. Also, for a target node, nodes with important high-order transitive relations will appear near the target node, while nodes with irrelevant co-occurrence relations will appear far away from the target node.

Recent research reveals that a common embedding method that only preserves the connection patterns of a graph can be effective [36]. In this paper, we employ LINE [17], which can preserve both the local and global network structures, as our embedding method to map the user and item relational graphs to their corresponding latent continuous spaces. Note that, one can employ or redesign other embedding methods to create suitable latent spaces, such as struc2vec [37] and DeepWalk [18].

Fig. 2b1) and (b2) are examples of the latent space after mapping when the space dimensions $l_u = l_v = 2$. Although there is a co-occurrence relation between user d and user e, their distance in the latent space may be far away due to the small weight of their edge in the user relational graph.

3.2.3. Step 3: construction of relational data

Based on the latent spaces, users' and items' relational data $\mathcal{N}_u$ and $\mathcal{N}_v$ can be constructed. Specifically, user u_a's relational data $\mathcal{N}_u(a)$ is a user set which contains K_u (a pre-defined hyper-parameter) nearest neighbors in the user latent space based on the particular distance metric in the space. The construction of item neighbors is similar to the user. For instance, item v_i's neighbors are defined as $\mathcal{N}_v(i)$ which contains top-K_v nearest neighbors items in the latent space to v_i. In this way, constructed relational data not only contains nodes with important high-order transitive relations, but also filters some nodes with irrelevant co-occurrence relations. Compared with only accounting for co-occurrence relations, our method reveals an in-depth understanding of potential user-user and item-item relations.

Fig. 2c1) and (c2) show examples of constructed relational data for user u_d and item v_a. The neighbor set for user u_d is $\mathcal{N}_u(d) = \{u_a, u_b, u_c\}$ when $K_u = 3$, and the neighbor set for item v_a is $\mathcal{N}_v(a) = \{v_b, v_c\}$ when $K_v = 2$.

3.2.4. Complexity analysis

Since we utilize LINE as our embedding method, the overall time complexity of relational graph mapping is $O(l_u \cdot S_u \cdot |\mathcal{E}_u| + l_v \cdot S_v \cdot |\mathcal{E}_v|)$, where S_u and S_v is the number of negative samples for users and items [17]. To obtain the nearest neighbors, the time complexity for a user is $O(M \cdot l_u)$. Similarly, for an item, the time complexity is $O(N \cdot l_v)$. In practice, we can utilize some acceleration computation methods proposed by previous works [38,39] to speed up the process of obtaining implicit neighbors. Note that the relational data $\mathcal{N}_u$ and $\mathcal{N}_v$ can be computed offline in advance, so we can prepare $\mathcal{N}_u$ and $\mathcal{N}_v$ before generating recommendations. In this work, the construction of the relational data $\mathcal{N}_u$ and $\mathcal{N}_v$ is constrained to utilizing the user-item interaction records in the training split.

3.3. Recommendation framework

In this subsection, we present the recommendation framework of the IRec, as illustrated in Fig. 3. By taking a user u_a, an item v_i and their neighbors $\mathcal{N}_u(a)$ and $\mathcal{N}_v(i)$ as inputs, the framework outputs the predicted rating $\hat{y}_{ai}$ from u_a to v_i. The recommendation framework consists of three parts: the embedding layer, the aggregation layer, and the prediction layer. Details of each part are described in the following.

3.3.1. Embedding layer

The embedding layer transforms the primitive features of users and items (e.g., ID, user gender, item category etc.) into low-dimensional dense vectors called *embeddings*. Similar to mainstream embedding based recommender models [7,8], we use one-hot vectors $\mathbf{u}_a^{id} \in \mathbb{R}^{M\times 1}$ and $\mathbf{v}_i^{id} \in \mathbb{R}^{N\times 1}$ to encode the ID features of user u_a and item v_i, respectively. We can obtain u_a's embedding $\mathbf{u}_a$ and v_i's embedding $\mathbf{v}_i$, as follows:

$$\mathbf{u}_a = \mathbf{U}^\top \cdot \mathbf{u}_a^{id}, \tag{3}$$

$$\mathbf{v}_i = \mathbf{V}^\top \cdot \mathbf{v}_i^{id}, \tag{4}$$

where $\mathbf{U} \in \mathbb{R}^{M\times d}$ and $\mathbf{V} \in \mathbb{R}^{N\times d}$ are the embedding matrices for the user features and item features, respectively. Here d is the dimension size of the embeddings.

3.3.2. Aggregation layer

Aggregation is a key component in the framework because the user and item representations are bound up with the implicit neighbors by aggregation. By taking related inputs $u_a, v_i, \mathcal{N}_u(a)$ and $\mathcal{N}_v(i)$, we design an aggregator to update u_a's and v_i's feature representations as follows:

$$\mathbf{u}_a^* = \mathrm{Agg}_u(u_a, \mathcal{N}_u(a)) = \mathrm{agg}_u^{(h)}(\mathbf{u}_a, \mathrm{agg}_u^{(l)}(\mathcal{N}_u(a))), \tag{5}$$

$$\mathbf{v}_i^* = \mathrm{Agg}_v(v_i,\ \mathcal{N}_v(i)) = \mathrm{agg}_v^{(h)}(\mathbf{v}_i, \mathrm{agg}_v^{(l)}(\mathcal{N}_v(i))), \tag{6}$$

where $\mathbf{u}_a$ and $\mathbf{v}_i$ are the representations for user u_a and item v_i from their embedding tables $\mathbf{U}$ and $\mathbf{V}$. $\mathrm{Agg}_.$ is a compound aggregation function used to update user and item representations. $\mathrm{agg}_.^{(l)}$ is the low-level aggregation function which maps the user (or item) neighbor set into a single embedding vector, and $\mathrm{agg}_.^{(h)}$ is the high-level aggregation function which integrates the target user (or item) representation and the neighbor representation into a new representation of the target user (or item).

Specifically, we present the implementation of the function $\mathrm{Agg}_.$. We illustrate the process for users and the same process works for items. We first compute the score π_{ab} between user u_a and one neighbor $u_b \in \mathcal{N}_u(a)$ via the attention mechanism as:

$$\pi_{ab} = (\mathbf{u}_a \odot \mathbf{u}_b)^\top \tanh(\mathbf{w}_{u(l)} \cdot [\mathbf{u}_b || \mathbf{v}_i] + \mathbf{b}_{u(l)}), \tag{7}$$

where $||$ and $\odot$ mean concatenation operation and element-wise product between two vectors. $\mathbf{w}_{u(l)}$ and $\mathbf{b}_{u(l)}$ are parameters of the attention mechanism in the user low-level aggregation function. We also employ tanh as the nonlinear activation function. In general, π_{ab} characterizes the importance of one neighbor u_b for the target user u_a. The intuition is as follows: the first term calculates the compatibility between user u_a and her neighbor u_b, and the second term computes the opinions of neighbor u_b on the target item v_i. Here, we simply employ the inner product on the two terms, however one can design a more sophisticated attention mechanism.

We then implement $\mathrm{agg}_u^{(l)}$ to characterize user u_a's implicit neighbors by the linear combination:

$$\mathbf{u}_{\mathcal{N}_u(a)} = \mathrm{agg}_u^{(l)}(\mathcal{N}_u(a)) = \sum_{u_b \in \mathcal{N}_u(a)} \tilde{\pi}_{ab} \cdot \mathbf{u}_b, \tag{8}$$

where $\tilde{\pi}_{ab}$ denotes the normalized attention coefficient:

$$\tilde{\pi}_{ab} = \frac{\exp(\pi_{ab})}{\sum\limits_{u_{b'} \in \mathcal{N}_u(a)} \exp(\pi_{ab'})}. \tag{9}$$

For the high-level function $\mathrm{agg}_u^{(h)}$, it aggregates the user representation $\mathbf{u}_a$ and its neighbor representations $\mathbf{u}_{\mathcal{N}_u(a)}$ as the new representation of user u_a. We implement $\mathrm{agg}_u^{(h)}$ by performing the summation operation between two representation vectors before employing nonlinear transformation:

$$\mathbf{u}_a^* = \mathrm{agg}_u^{(h)}(\mathbf{u}_a, \mathbf{u}_{\mathcal{N}_u(a)}) = \sigma(\mathbf{w}_{u(h)} \cdot (\mathbf{u}_a + \mathbf{u}_{\mathcal{N}_u(a)}) + \mathbf{b}_{u(h)}), \tag{10}$$

where $\mathbf{w}_{u(h)}, \mathbf{b}_{u(h)}$ are parameters in the user high-level aggregation function and σ is the nonlinear activation function. We try various kinds of operations, such as concatenation, and find the summation operation always shows the best performance.

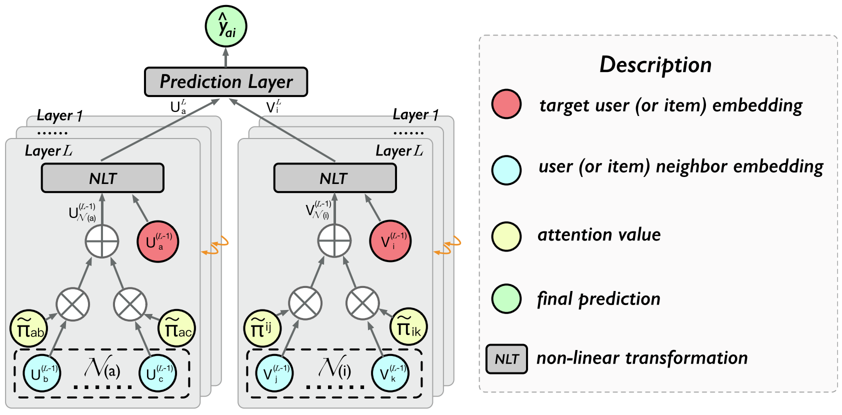


Fig. 3. IRec's recommendation framework's architecture.

Through a single aggregation layer, user (or item) representation is dependent on itself as well as the direct neighbors. We can further stack more layers to obtain high-order information from the multi-hop neighbors of users (or items). More formally, in the l-th layer, for user u_a and item v_i, their representations are defined as:

$$\mathbf{u}_a^l = \mathrm{agg}_u^{(h)}(\mathbf{u}_a^{(l-1)}, \mathbf{u}_{\mathcal{N}_u(a)}^{(l-1)}), \tag{11}$$

$$\mathbf{v}_i^l = \mathrm{agg}_v^{(h)}(\mathbf{v}_i^{(l-1)}, \mathbf{v}_{\mathcal{N}_v(i)}^{(l-1)}), \tag{12}$$

where $\mathbf{u}_{\mathcal{N}_u(a)}^{(l-1)}$ and $\mathbf{v}_{\mathcal{N}_v(i)}^{(l-1)}$ are defined as:

$$\mathbf{u}_{\mathcal{N}(a)}^{(l-1)} = \sum_{u_b \in \mathcal{N}_u(a)} \tilde{\pi}_{ab} \cdot \mathbf{u}_b^{(l-1)}, \tag{13}$$

$$\mathbf{v}_{\mathcal{N}(i)}^{(l-1)} = \sum_{v_j \in \mathcal{N}_v(i)} \tilde{\pi}_{ij} \cdot \mathbf{v}_j^{(l-1)}. \tag{14}$$

3.3.3. Prediction layer

After the L aggregation layer, we feed user representation $\mathbf{u}_a^L$ and item representation $\mathbf{v}_i^L$ into a function $p : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ for rating prediction.

$$\hat{y}_{ai} = p(\mathbf{u}_a^L, \mathbf{v}_i^L). \tag{15}$$

Here we implement the prediction function p as the MLP component [7], which can model complicated interactions between users and items. Specifically, the MLP component is implemented with two hidden layers (tower structure: $2d - d - 1$) as:

$$\hat{y}_{ai} = \mathbf{w}_p^2 \cdot \sigma\left(\mathbf{w}_p^1 \cdot [\mathbf{u}_a^L || \mathbf{v}_i^L] + \mathbf{b}_p^1\right) + \mathbf{b}_p^2, \tag{16}$$

where $\mathbf{w}_{\cdot}$ and $\mathbf{b}_{\cdot}$ denote the weight matrix and bias parameters in the MLP. One can utilize various prediction functions, such as inner product, to generate recommendations.

3.3.4. Learning algorithm

To estimate the parameters of the recommendation framework, we have the following objective function:

$$\min \mathcal{L} = \mathcal{L}_{\mathrm{Rec}} + \lambda ||\Theta||^2, \tag{17}$$

where $\mathcal{L}_{\mathrm{Rec}}$ measures the loss in the recommendation framework. Our paper is centered on the rating prediction for recommendation which is a regression problem. For the regression, we formulate $\mathcal{L}_{\mathrm{Rec}}$ as the squared loss:

$$\mathcal{L}_{\mathrm{Rec}} = \frac{1}{|\mathcal{O}|} \sum_{(a,i) \in \mathcal{O}} (y_{ai} - \hat{y}_{ai})^2, \tag{18}$$

where $\mathcal{O}$ denotes the observed ratings in $\mathbf{Y}$.

The second term in Eq. (17) is the L2 regularization term to control model complexity and to avoid over-fitting. $\Theta = \left\{\mathbf{U}, \mathbf{V}, \mathbf{w}_p^1, \mathbf{w}_p^2, \mathbf{w}_{u(l)}^l, \mathbf{w}_{u(h)}^l, \mathbf{w}_{v(l)}^l, \mathbf{w}_{v(h)}^l, \forall l \in \{1, \cdots, L\}\right\}$ is the parameter set in the framework.

The training process of the recommendation framework is summarized in Algorithm 1.

Algorithm 1 Training algorithm for the recommendation framework

```
Input: Interaction matrix $\mathbf{Y}$; user neighbor set $\mathcal{N}_u$; item neighbor set $\mathcal{N}_v$; balancing factors $\lambda$; learning rate $\eta$
Output: Prediction function $\mathcal{F}(u, v|\Theta, \mathbf{Y}, \mathcal{N}_u, \mathcal{N}_v)$
1: Initialize all parameters in $\Theta$
2: repeat
3:   Sample a minibatch of user-item interaction data from $\mathbf{Y}$
4:   Calculate $\mathcal{L} \leftarrow \mathcal{L}_{\text{Rec}} + \lambda||\Theta||^2$
5:   for each parameter $\vartheta \in \Theta$
6:     Calculate $\partial\mathcal{L}/\partial\vartheta$ on the minibatch by backpropagation
7:     Update $\vartheta$ by gradient descent with learning rate $\eta$
8:   end for
9: until $\mathcal{L}$ converges or is sufficiently small
10: return $\mathcal{F}(u, v|\Theta, \mathbf{Y}, \mathcal{N}_u, \mathcal{N}_v)$
```

3.3.5. Complexity analysis

In this subsection, we discuss the time complexity and space complexity for the recommendation framework.

Time complexity. The time cost of the recommendation framework mainly comes from the aggregation layer. For users, the matrix multiplication in the aggregation layer has computational complexity $O(M \cdot K_u \cdot L \cdot d^2)$, where M is the number of users, K_u is the number of neighbors for each user, L is the total layers in the aggregation layer and d denotes the embedding size. Similarly, the time consumption for items in the aggregation layer is $O(N \cdot K_v \cdot L \cdot d^2)$, where N is the number of items and K_v is the number of neighbors for each item. In general, the overall training complexity is $O(M \cdot K_u \cdot L \cdot d^2 + N \cdot K_v \cdot L \cdot d^2)$. In fact, as shown in our experiment section, the framework reaches the best performance when $L = 1$. Also, the number of neighbors for users and items are limited with $K_u = K_v \ll \min\{M, N\}$. Therefore, the total time complexity of the recommendation framework in IRec is acceptable.

Space complexity. As shown in Eq. (17), the model parameters Θcomprise two parts: embedding tables for users and items $\Theta_1 = \{\mathbf{U}, \mathbf{V}\}$, and weight parameters in neural components $\Theta_2 = \left\{\mathbf{w}_p^1, \mathbf{w}_p^2, \mathbf{w}_{u(l)}^l, \mathbf{w}_{u(h)}^l, \mathbf{w}_{v(l)}^l, \mathbf{w}_{v(h)}^l, \forall l \in \{1, \cdots, L\}\right\}$. Parameter set Θ_1 is identical to that of the classical embedding-based models, such as MF [6], BPR [40]. Parameter set Θ_2 is lighter than Θ_1 and can be neglected because (i) the parameters in Θ_2 are shared by all users and items; (ii) the dimension of each parameter in Θ_2 is far less than the number of users and items. Therefore, the space complexity of the framework is the same as the classical embedding models.

4. Experiments

In this section, we evaluate our method IRec in four real-world scenarios: movie, business, book, and restaurant recommendations. We first introduce the experiment settings, then present the experiment results. We also analyze the choice of hyper-parameters, the training efficiency, and some case studies in this section.

4.1. Experiment setup

In this subsection, we introduce the datasets, baselines, evaluation protocols, and the choice of hyper-parameters.

4.1.1. Datasets

Four datasets DVD[1] movie dataset, Yelp[2] business dataset, Douban[3] book dataset, and Dianping[4] restaurant dataset are used in our experiments. Each dataset contains users' ratings (ranging from 1 to 5) on the items. The statistics of the datasets are summarized in Table 1.

4.1.2. Baselines

To verify the performance of our proposed method IRec, we compared it with the following state-of-art recommendation methods. The characteristics of the comparison methods are listed as follows:

[1] DVD:https://www.librec.net/datasets.html.
[2] Yelp:http://www.yelp.com/.
[3] Douban:https://www.douban.com/.
[4] Dianping:https://www.dianping.com/.

Table 1
Basic statistics for the four datasets: Movie (DVD), Business (Yelp), Book (Douban), and Restaurant (Dianping).

dataset	# users	# items	# interactions	density
Movie	2,433	12,838	32,893	0.105%
Business	10,580	13,870	171,102	0.117%
Book	11,777	20,697	190,590	0.078%
Restaurant	10,549	17,707	188,813	0.101%

- **SVD++** is a well-known baseline, which is a hybrid model combining the latent factor model and the neighborhood model [13].
- **NFM** is a feature-based factorization model, which improves FM [41] by using the MLP component to capture the high-order feature interaction [26]. Here we concatenate user ID embeddings and item ID embeddings as input for NFM.
- **GCMC** is a graph-based recommendation framework which adopts a graph auto-encoder in a user-item bipartite graph to learn user and item embeddings for rating prediction [12].
- **NGCF** is a state-of-the-art graph-based recommender system which utilizes multiple propagation layers to learn user and item representations by propagating embeddings on the user-item bipartite graph [11]. For the rating prediction task, we replace the inner product with a two-layer MLP component in the prediction layer to enhance its performance.
- **CUNE** is a semantic social recommendation method which identifies semantic social friends from the collaborative user network and models these semantic relations as regularization terms to constrain the matrix factorization model [42]. In particular, they do not consider the neighbor information of items.
- **CMN** is a state-of-the-art memory-based model which designs the memory slots of similar users to learn user embeddings [10]. Note that it only focuses on the user's neighbors without accounting for the information about similar items.
- **MMCF** is another state-of-the-art memory-based model which models user-user and item-item co-occurrence contexts by memory networks [8]. It is the work which is most related to us. Different from our methods, it only focuses on co-occurrence relations and ignores high-order transitive relations among users and items.

4.1.3. Evaluation protocols

Three classes of metrics are adopted to evaluate the recommendation quality:

(i) For the rating prediction task, we utilize mean absolute error (MAE) and root mean square error (RMSE) as the performance metrics, which are widely adopted in many related works [12,13,26,43]. Smaller values of MAE and RMSE indicate a better recommendation, which are defined as follows

$$\mathrm{MAE} = \frac{1}{|\mathcal{D}_{\mathrm{test}}|}\sum_{(a,i)\in\mathcal{D}_{\mathrm{test}}} |y_{ai} - \hat{y}_{ai}|, \tag{19}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{|\mathcal{D}_{\mathrm{test}}|}\sum_{(a,i)\in\mathcal{D}_{\mathrm{test}}} (y_{ai} - \hat{y}_{ai})^2}, \tag{20}$$

where $\mathcal{D}_{test}$ denotes the test set of the rating records.

(ii) We further adopt three rank-based metrics precision, recall and F1-score to evaluate different methods. Following the approaches in [44], precision, recall and F1-score are defined as follows:

$$\mathrm{Precision} = \frac{1}{M}\sum_{i=1}^{M}\frac{|\mathrm{Fav}(i)\cap\mathrm{Rec}(i)|}{|\mathrm{Rec}(i)|}, \tag{21}$$

$$\mathrm{Recall} = \frac{1}{M}\sum_{i=1}^{M}\frac{|\mathrm{Fav}(i)\cap\mathrm{Rec}(i)|}{|\mathrm{Fav}(i)|}, \tag{22}$$

$$\mathrm{F1-score} = \frac{2\times\mathrm{Precision}\times\mathrm{Recall}}{\mathrm{Precision}+\mathrm{Recall}}, \tag{23}$$

where $Fav(i) = \{j \in \Omega(i) \mid y_{ij} \geqslant 4\}$ and $\Omega(i)$ is the item set that user i has interacted with in the test set. Since the users' ratings on the items range from 1 to 5, a rating of "4" or "5" usually indicates that the users like the items. Therefore, we define $Fav(i)$ is the favorite item set of user i. $Rec(i) = \{j \in \Omega(i) \mid b(\hat{y}_{ij}) \geqslant 4\}$ is the set of items which will be recommended to user i and $b(\cdot)$ is the rounding function, which rounds the predicted rating $\hat{y}_{ij}$ to an integer rating. The bigger the precision, recall, and F1-score values, the better the ranking.

(iii) In addition, we also use the three top-k based ranking metrics in our experiment: Precision@k (Pre@k for short), Recall@k (Rec@k for short), F1@k. Following the approaches in [44], three metrics are defined as follows:

$$\mathrm{Pre@}k = \frac{1}{M}\sum_{i=1}^{M}\mathrm{Pre@}k(i) = \frac{1}{M}\sum_{i=1}^{M}\frac{|Fav(i)\cap\omega(i)|}{k}, \tag{24}$$

$$\mathrm{Rec@}k = \frac{1}{M}\sum_{i=1}^{M}\mathrm{Rec@}k(i) = \frac{1}{M}\sum_{i=1}^{M}\frac{|Fav(i)\cap\omega(i)|}{|Fav(i)|}, \tag{25}$$

$$\mathrm{F1@k} = \frac{2\times\mathrm{Pre@k}\times\mathrm{Rec@k}}{\mathrm{Pre@k}+\mathrm{Rec@k}}, \tag{26}$$

where $\omega(i)$ is the top k item set in the ranking list determined by the trained models for user i.

4.1.4. Parameter settings

For the neighbor construction phase, we define $l_u = l_v = 8, S_u = S_v = 5$ and utilize Euclidean distance to calculate the distance in the latent spaces. We implemented the recommendation framework of IRec with Tensorflow which is a Python library for deep learning. For each dataset, we randomly split it into training, validation, and test sets following the 6: 2: 2 ratio. We repeated each experiment 5 times and reported the average performance. The framework parameters are first initialized by the Xavier initializer [45], and then updated them by conducting mini-batch Adam. For the selection of the activation function, we utilized LeakyReLU by default.

For our framework, there are six key hyper-parameters that need to be tuned, including the dimension of embeddings d, layer size L, number of neighbors K, batch size b, balancing factor λ and learning rate η. The hyper-parameters were tuned on the validation set using grid search which is widely used in many deep models [11,16,25]. Table 2 shows our hyper-parameter settings. The key hyper-parameter settings for the baselines are defined as follows. For NFM, we utilize a one-layer MLP component according to the original paper [26]. Regarding NGCF, we tune the depth of layer L between $\{1,2,3,4\}$, and find NGCF performs best with $L = 2$ for the movie and book datasets, and $L = 3$ for the business and restaurant datasets. For CMN and MMCF, the memory hop H is tuned between $\{1,2,3,4\}$, and we find $H = 2$ reaches the best performance. The settings for the other hyper-parameters for all the baselines are reached by either empirical study or following the original papers.

Deep models have a strong representation ability but they usually suffer from the over-fitting problems. To prevent over-fitting, we adopt L2 regularization (as mentioned in subSection 3.3.4) and the early stopping strategy (i.e., premature stopping if RMSE on the validation set does not increase for 3 successive epochs). Fig. 4 shows the training and validation error of each epoch of IRec. From the figure, we can see that 20 epochs are sufficient for our method to train and converge. If the model continues learning, then a situation of over-fitting will occur (i.e., the validation loss has begun to increase). We also tried the dropout technique, and found that introducing dropout masks slightly decreases the performance. A possible rea-

Table 2
Hyper-parameter settings for the four datasets: Movie (DVD), Business (Yelp), Book (Douban), and Restaurant (Dianping).

Dataset	Hyper-parameter settings					
Movie	$d = 8$	$L = 1$	$K = 6$	$b = 256$	$\lambda = 10^{-4}$	$\eta = 10^{-2}$
Business	$d = 16$	$L = 1$	$K = 6$	$b = 1024$	$\lambda = 10^{-4}$	$\eta = 2\times10^{-2}$
Book	$d = 16$	$L = 1$	$K = 5$	$b = 1024$	$\lambda = 2\times10^{-4}$	$\eta = 10^{-2}$
Restaurant	$d = 16$	$L = 1$	$K = 8$	$b = 1024$	$\lambda = 10^{-4}$	$\eta = 2\times10^{-2}$

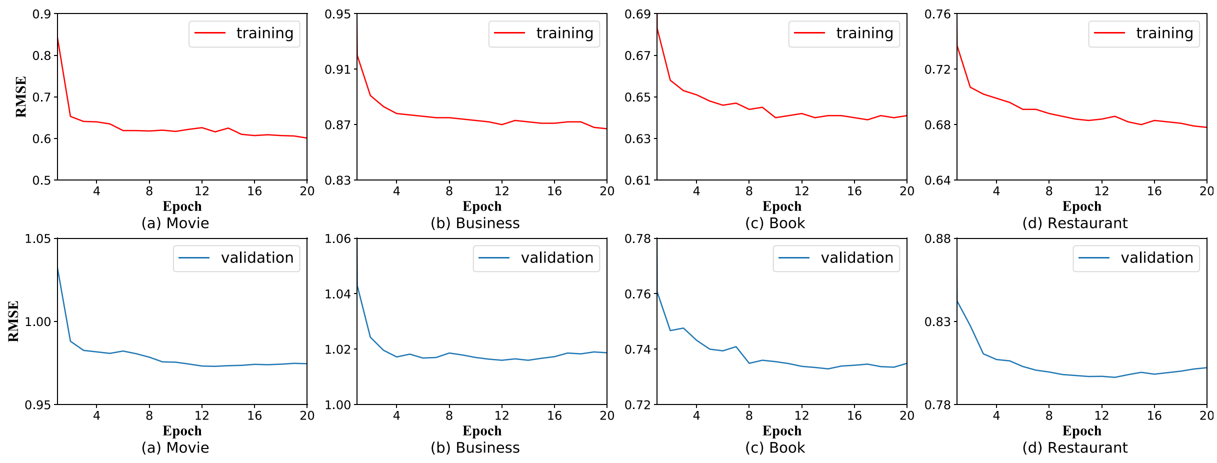


Fig. 4. Training and validation error of each epoch of IRec on the four datasets.

son for this is that we do not introduce too many weight parameters with large dimensions in neural components, thus the dropout technique may not be as helpful to our model as it is to other deep and large models. Therefore, we do not introduce dropout mechanisms.

4.2. Empirical study

We conduct an empirical study to investigate the correlation between users (or items) and their co-occurrence neighbors. To formulate this issue, we utilize the difference in the explicit feedback (i.e. ratings) as the index. Taking users for example, if a co-occurrence user pair had engaged a common item before and the difference in their ratings on the item is greater than or equal to 2 (in a 5-star system), we believe this co-occurrence user pair encodes irrelevant (or noisy) information. To this end, we make statistics on the four datasets used in this paper. The results are presented in Fig. 5. We observed that both the user co-occurrence relation and item co-occurrence relation encode irrelevant (or noisy) information in the four datasets. In particular, more than 1/5 of the user co-occurrence relations and 1/4 of the item co-occurrence relations in the business dataset exist such cases. The above findings empirically demonstrate that not all co-occurrence relations contain useful information, so it is important to filter out irrelevant information to construct meaningful implicit neighbors for users and items.

4.3. Performance comparison

Table 3 and Figs. 6–8 show the performance of all the compared methods on the four datasets. From the results, we make the following main observations:

(i) SVD++ achieves poor performance on the four datasets, which indicates that shallow representation is insufficient to capture complex user-item interactions. NFM consistently outperforms SVD++, which suggests the significance of non-linear feature interactions between user and item embeddings in recommender systems. However, both SVD++ and NFM ignore user-user and item-item relations.
(ii) Both SVD++ and CUNE are shallow models based on matrix factorization, while CUNE achieves better performance than SVD++; meanwhile, for the deep recommendation models, CMN and MMCF generally achieve better performance than NFM in most cases. These results suggest that considering potential user-user and item-item neighbors can enhance the recommendation performance. In addition, MMCF consistently outperforms CUNE and CMN. This makes sense since CUNE and CMN only account for user neighbor information, while MMCF considers co-occurrence information for both users and items.
(iii) For GNN-based models, NGCF achieves better performance than GCMC in most cases. The reason might be because GCMC only incorporates first-order neighbors for users and items in the bipartite graph, while NGCF models the high-order information.
(iv) Our method IRec consistently yields the best performance on the four datasets, which demonstrates the effectiveness of IRec on rating prediction and top-k recommendation. We also conduct one-sample t-tests and $p < 0.05$ indicates that the improvements of IRec over the best baseline are statistically significant.

4.4. Data sparsity and cold start issues

As mentioned in many studies in the literatures [32,44,46,47], data sparsity and cold start are two challenges faced by most recommenders. In this subsection, we investigated the ability of our model to handle these two issues.

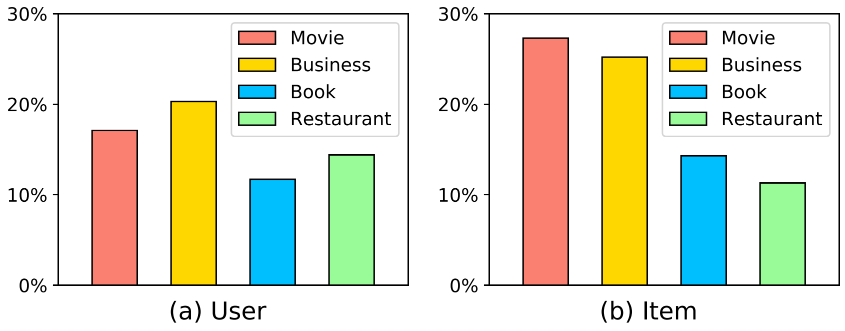


Fig. 5. Empirical study on the four datasets. (a) Percentage of co-occurrence user pairs encoding irrelevant (or noisy) information. (b) Percentage of co-occurrence item pairs encoding irrelevant (or noisy) information.

Table 3
Recommendation performance of seven competing methods and our method IRec on the four datasets. The proposed method IRec achieves the best performance on all metrics, as shown in boldface. * denotes the statistical significance for $p < 0.05$ compared to the best baseline.

Movie	SVD++	NFM	GCMC	NGCF	CUNE	CMN	MMCF	IRec
MAE	0.8183	0.8178	0.7790	0.7893	0.8181	0.8097	0.7804	**0.7594***
RMSE	1.0503	1.0391	1.0177	1.0120	1.0406	1.0192	1.0068	**0.9876***
Precision	0.8715	0.8908	0.8940	0.9027	0.8918	0.8733	0.8971	**0.9166***
Recall	0.7386	0.8285	0.8197	0.7990	0.7985	0.8270	0.8640	**0.8757***
F1-score	0.7996	0.8585	0.8552	0.8477	0.8426	0.8495	0.8802	**0.8957***
Business	SVD++	NFM	GCMC	NGCF	CUNE	CMN	MMCF	IRec
MAE	0.8318	0.8045	0.8167	0.7995	0.8035	0.8122	0.8032	**0.7889***
RMSE	1.0469	1.0357	1.0437	1.0266	1.0351	1.0348	1.0344	**1.0164***
Precision	0.8187	0.8196	0.8507	0.8509	0.8514	0.8302	0.8349	**0.8569***
Recall	0.6330	0.6538	0.6697	0.6769	0.6490	0.7022	0.7264	**0.7361***
F1-score	0.7140	0.7274	0.7494	0.7540	0.7365	0.7609	0.7769	**0.7919***
Book	SVD++	NFM	GCMC	NGCF	CUNE	CMN	MMCF	IRec
MAE	0.6079	0.5909	0.6040	0.5958	0.5960	0.5947	0.5926	**0.5881***
RMSE	0.7648	0.7436	0.7549	0.7523	0.7442	0.7419	0.7387	**0.7331***
Precision	0.8674	0.8820	0.8760	0.8835	0.8812	0.8571	0.8799	**0.8856***
Recall	0.7122	0.7902	0.8168	0.7853	0.7639	0.7911	0.8137	**0.8494***
F1-score	0.7822	0.8336	0.8454	0.8315	0.8184	0.8228	0.8455	**0.8671***
Restaurant	SVD++	NFM	GCMC	NGCF	CUNE	CMN	MMCF	IRec
MAE	0.6640	0.6459	0.6480	0.6263	0.6365	0.6414	0.6238	**0.6207***
RMSE	0.8682	0.8488	0.8385	0.8239	0.8356	0.8473	0.8198	**0.8077***
Precision	0.8491	0.8460	0.8669	0.8746	0.8495	0.8524	0.8654	**0.8750***
Recall	0.6907	0.7058	0.7156	0.7460	0.7404	0.7084	0.7252	**0.7845***
F1-score	0.7618	0.7696	0.7840	0.8052	0.7912	0.7738	0.7891	**0.8273***

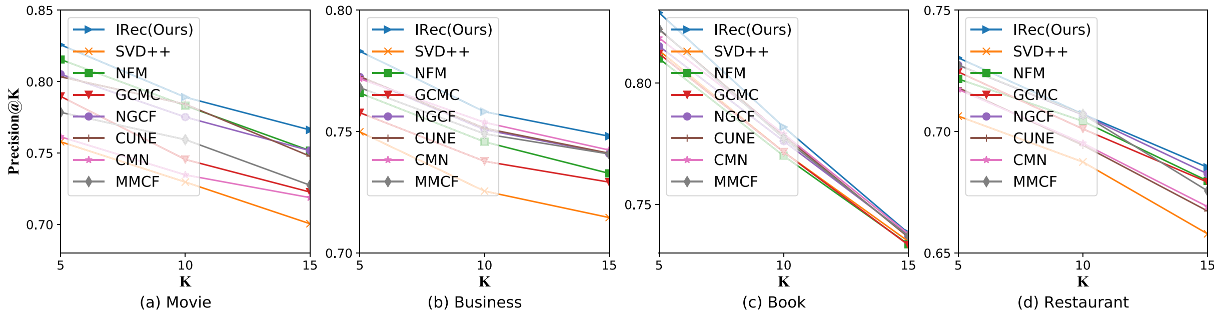


Fig. 6. The results of Precision@K on the four datasets.

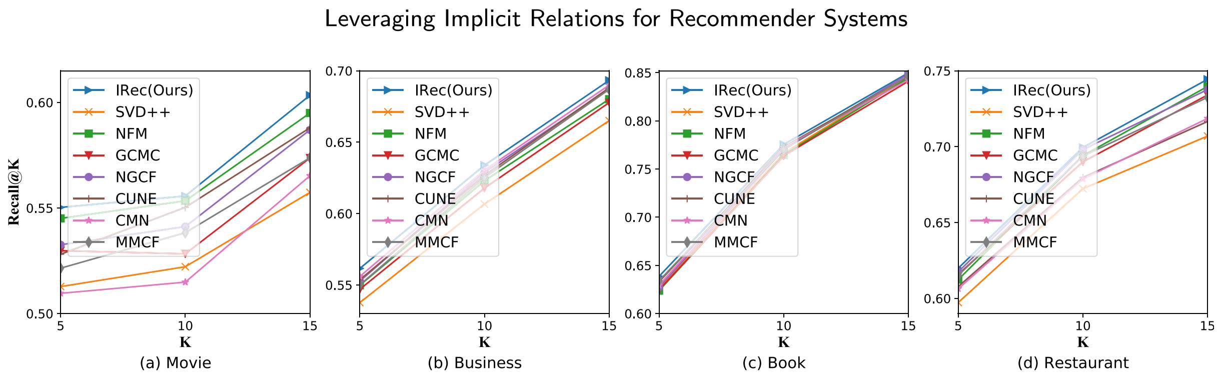


Fig. 7. The results of Recall@K on the four datasets.

4.4.1. Results in data sparse scenarios

The data sparsity problem is a great challenge for most recommender systems. To investigate the effect of data sparsity, we divide the test users into four groups with different sparsity levels based on the number of observed ratings in the train-

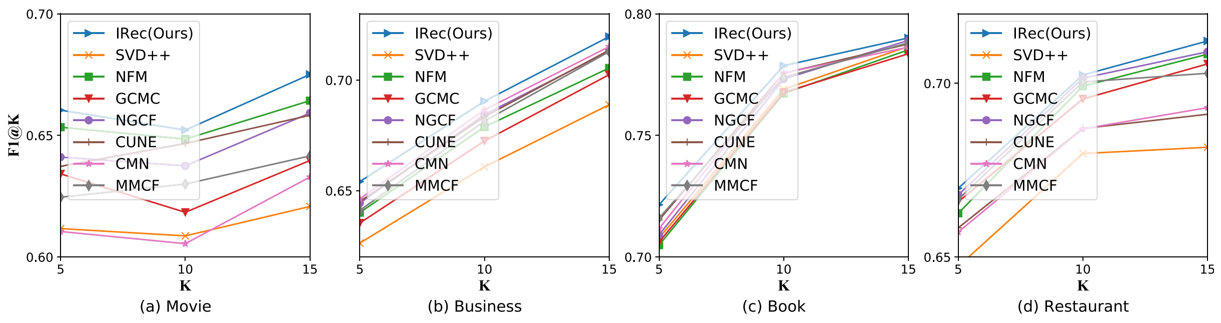


Fig. 8. The results of F1@K on the four datasets.

ing data, and keep each group including a similar number of interactions. For example, [10,29) in the Movie dataset means each user in this group has at least 10 rating records and less than 29 rating records. Fig. 9 shows the RMSE results for the different user groups with different models on the four datasets. From the results, we observe that our IRec outperforms the other methods in most cases. It is worth mentioning that IRec consistently outperforms all baselines in the first group on the four datasets, which verifies that our method IRec can maintain a good performance when data are extremely sparse.

4.4.2. Results in cold start scenarios

We consider addressing two cold start scenarios, namely the cold-start user problem and cold-start item problem. We treat those who have rated x or fewer ratings as cold start users and those that have been rated less than x as cold start items. Following other work [44], we set $x = 5$. Tables 4 and 5 illustrate the RMSE results of our method IRec and the other baselines in two cold-start scenarios on four datasets. In the tables, * denotes the statistical significance for $p < 0.05$, compared to the best baseline. We can see that our method IRec is beneficial to relatively inactive users and items in four recommendation scenarios.

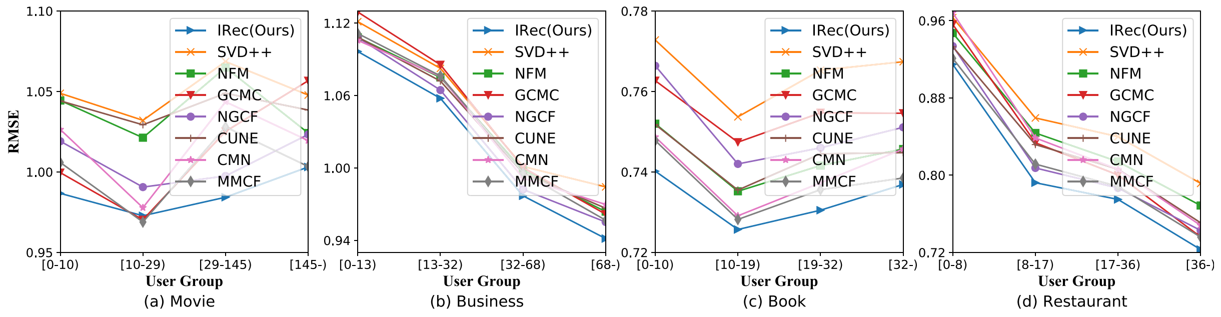


Fig. 9. Performance comparison over the sparsity distribution of user groups on the four datasets.

Table 4
RMSE results on testing cold start users on four datasets. The proposed method IRec achieves best performances on all metrics which are in boldface. * denotes the statistical significance for $p < 0.05$, compared to the best baseline.

Method	SVD++	NFM	GCMC	NGCF	CUNE	CMN	MMCF	IRec
Movie	1.0617	1.0525	1.0465	1.0336	1.0516	1.0424	1.0272	**0.9980***
Business	1.0880	1.0987	1.0851	1.0743	1.0696	1.0778	1.0643	**1.0399***
Book	0.7794	0.7713	0.7690	0.7621	0.7752	0.7700	0.7591	**0.7409***
Restaurant	0.9824	0.9798	0.9792	0.9533	0.9605	0.9572	0.9443	**0.9325***

Table 5
RMSE results on testing cold start items on four datasets. The proposed method IRec achieves best performances on all metrics which are in boldface. * denotes the statistical significance for $p < 0.05$, compared to the best baseline.

Method	SVD++	NFM	GCMC	NGCF	CUNE	CMN	MMCF	IRec
Movie	1.0861	1.0705	1.0446	1.0308	1.0574	1.0333	1.0205	**0.9959***
Business	1.1676	1.1826	1.1539	1.1344	1.1612	1.1563	1.1440	**1.1213***
Book	0.7981	0.7873	0.7880	0.7942	0.7910	0.7943	0.7708	**0.7610***
Restaurant	0.8880	0.8958	0.9084	0.8876	0.8997	0.8929	0.8858	**0.8671***

4.5. Parameter sensitivity

We explore the impact of three hyper-parameters: embedding size d, neighbor size k, and number of layers L in the aggregation layer. The results on the Movie and Business datasets are plotted in Fig. 10. We make the following observations: (i) a proper embedding size d is needed. If d is too small, the model lacks expressiveness, while a too large d increases the complexity of the recommendation framework and may overfit the datasets. (ii) The performance is improved with an increase in neighbor size K at the beginning because more neighbors provide more information. However, the performance takes a downward trend when K is larger than 6, since too many neighbors may introduce noise which reduces the accuracy of the prediction. (iii) In relation to the number of layers L in the aggregation layer, we find that when $L = 1$, it is good enough because a larger L will bring massive noise when generating high-quality user and item representations and may lead to over-fitting. Similar results can be found in many other studies [16,48].

4.6. Efficiency analysis

In this section, we discuss the experiments to explore the training efficiency of our IRec and two related methods CMN and MMCF, which explicitly account for the user-user and item-item neighbor information for recommendation.

We first evaluate the training time of one iteration in the same environment (1.8 GHz Intel Core i5 and 8 GB of RAM memory). Three methods are executed with 20 iterations and we report the average runtime. Table 6 shows the computation time for the four datasets. We observe that IRec is 1.5–2 times faster than CMN and 4–5 times faster than MMCF in one iteration, which confirms that our IRec has better training efficiency.

We then compare the number of trainable parameters for CMN, MMCF, and IRec. Table 7 summarizes the number of parameters of each method on embedding size 16 on the four datasets. We observe that IRec needs the least trainable parameters compared with CMN and MMCF. Specifically, MMCF requires more than double the number of parameters compared with our method IRec. This demonstrates IRec is a light yet effective model for recommendation.

4.7. Co-occurrence relation vs implicit relation

In this subsection, we compare the co-occurrence relation and the implicit relation. To this end, we compare IRec with its variant CRec. CRec utilizes the recommendation framework of IRec and leverages the co-occurrence relation instead of the implicit relation for recommendation. Table 8 shows the performance of IRec and CRec on the four datasets. From the results, we find that the implicit relation can further improve the recommendation performance compared with the co-occurrence relation.

In the above experiments, we have validated the effectiveness of IRec which leverages the implicit relation for recommendation. Next, we investigate whether the co-occurrence relation and constructed implicit relation overlap with each other. For this purpose, we make statistics on the Movie dataset from the perspective of users and show an overlapping relationship between the co-occurrence user pairs and the implicit user pairs.

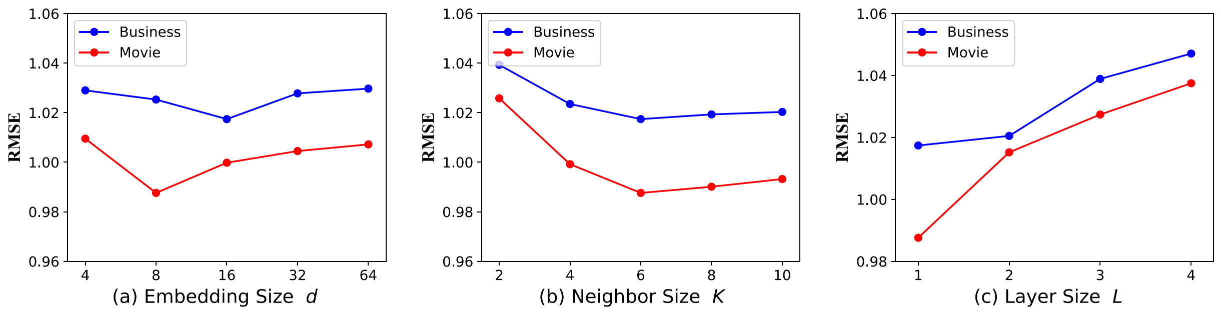


Fig. 10. Parameter sensitivity of IRec's recommendation framework on the Movie and Business datasets w.r.t. (a) embedding size d, (b) neighbor size K, and (c) number of layers L in the aggregation layer.

Table 6
Training time of one iteration of CMN, MMCF, and IRec on the four datasets. 's' denotes 'second'.

Method	Movie	Business	Book	Restaurant
CMN	0.87 s	7.59 s	8.83 s	7.59 s
MMCF	2.58 s	17.09 s	19.98 s	19.29 s
IRec	0.59 s	3.62 s	5.06 s	4.48 s

Table 7
Number of parameters of CMN, MMCF, and IRec on embedding size 16 on the four datasets. ‘k’ denotes ‘10^3’.

Method	Movie	Business	Book	Restaurant
CMN	300.7 k	577.9 k	725.4 k	638.3 k
MMCF	492.3 k	786.0 k	1042.8 k	907.8 k
IRec	245.9 k	392.7 k	521.1 k	453.6 k

Table 8
The results of MAE between IRec and its variant CRec on the four datasets.

Method	Movie	Business	Book	Restaurant
IRec	0.7594	0.7889	0.5881	0.6207
CRec	0.7784	0.7925	0.5971	0.6310

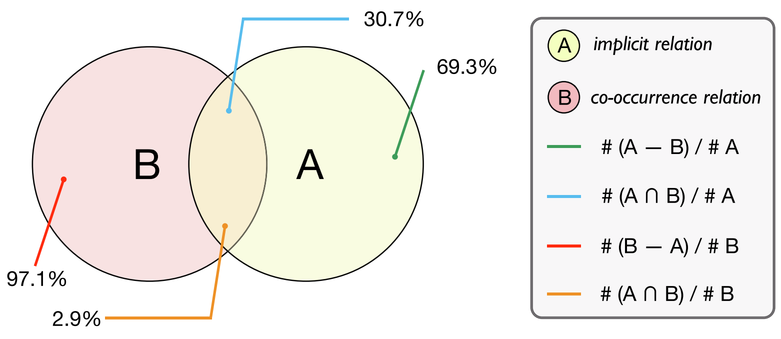


Fig. 11. Overlaping relationship between the co-occurrence user pairs and implicit user pairs.

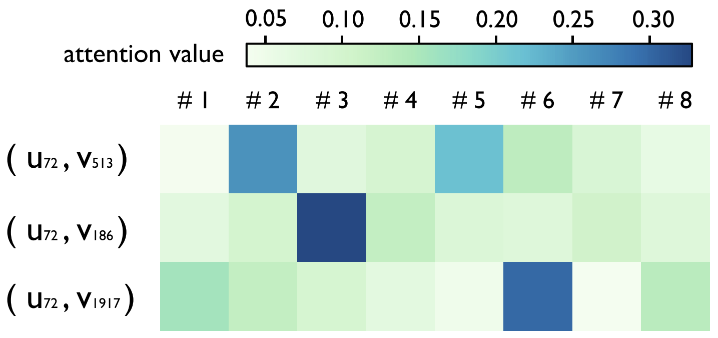


Fig. 12. Attention heatmap for the neighbors of three user-item pairs from the restaurant dataset.

From Fig. 11, we make the following findings: (i) only around 30% of implicit user pairs are co-occurrence relations. That is to say, if we only consider co-occurrence user pairs, a large portion of potentially relevant users is missed. (ii) Most co-occurrence user pairs have less relevance, which demonstrates the necessity of filtering the co-occurrence relation. Based on the above observations, we conclude that only accounting for the co-occurrence relations may result in a loss of useful information and it introduces some irrelevant information, while our method IRec leverages embedding methods to generate meaningful implicit relations for users and items.

4.8. Attention analysis

Benefiting from the attention mechanism, we can visualize the attention weights placed on the neighbors for users and items, which reflects how the model learns. In this subsection, we analyze the attention mechanism from the perspective of users to show the learning process of IRec’s recommendation framework and we obtain similar observations for items. We randomly selected one user u_{72} from the Restaurant dataset, and three relevant items $v_{513}, v_{186}, v_{1917}$ (from the test set). Fig. 12 shows the attention weights of the user u_{72}’s neighbors for the three user-item pairs. For convenience, we label the neighbor IDs starting from 1, which may not necessarily reflect the true ID from the dataset. From the heatmap, we make the following findings: (i) not all neighbors make the same contribution when generating recommendations. For instance, for the user-item pair (u_{72}, v_{513}), the attention weights of user u_{72}'s neighbor # 2 and # 5 are relatively high. The reason for this may be that neighbor # 2 and # 5 have rated item v_{513} in the training set. Therefore, neighbor # 2 and # 5 will provide more useful information when making recommendations. (ii) For different items, the attention distributions of the neighbors are different, which reflects the attention mechanism that can adaptively measure the influence strength of the neighbors.

5. Conclusion and future work

In this work, we proposed a method called IRec to leverage implicit neighbors for better recommendations. IRec includes (i) a neighbor construction method that utilizes the user-item interaction information to construct implicit neighbor sets for each user and item; and (ii) a novel framework that integrates constructed neighbor sets into the recommendation task. We conducted extensive experiments on four real-world datasets. The experiment results demonstrate the superiority of IRec over several state-of-the-art methods in rating prediction and top-*k* recommendation. The results also show that our method is beneficial for relatively inactive and cold-start users.

For future work, we will (i) integrate side information into IRec such as knowledge graphs and social networks to further enhance the recommendation; (ii) employ more embedding methods in the neighbor construction phase to dig out user-user and item-item relationships; and (iii) try to generate recommendation explanations to comprehend the user behaviors and item attributes.

CRediT authorship contribution statement

Anchen Li: Conceptualization, Methodology, Data curation, Formal analysis, Writing - original draft. **Bo Yang:** Methodology, Writing - review & editing, Funding acquisition, Supervision. **Huan Huo:** Writing - review & editing, Supervision. **Farookh Khadeer Hussain:** Writing - review & editing, Supervision.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant No. 61876069; Jilin Province Key Scientific and Technological Research and Development Project under Grant Nos. 20180201067GX and 20180201044GX; and Jilin Province Natural Science Foundation under Grant No. 20200201036JC.

References

[1] Y.-C. Lee, T. Kim, J. Choi, X. He, S.-W. Kim, M-bpr: a novel approach to improving bpr for recommendation with multi-type pair-wise preferences, Inf. Sci. 547 (2021) 255–270.
[2] F.S. de Aguiar Neto, A.F. da Costa, M.G. Manzato, R.J. Campello, Pre-processing approaches for collaborative filtering based on hierarchical clustering, Inf. Sci. 534 (2020) 172–191.
[3] G.R. Lima, C.E. Mello, A. Lyra, G. Zimbrao, Applying landmarks to enhance memory-based collaborative filtering, Inf. Sci. 513 (2020) 412–428.
[4] X.-Y. Huang, B. Liang, W. Li, Online collaborative filtering with local and global consistency, Inf. Sci. 506 (2020) 366–382.
[5] A. Li, B. Yang, H. Chen, G. Xu, Hyperbolic neural collaborative recommender, arXiv preprint arXiv:2104.07414.
[6] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (8) (2009) 30–37.
[7] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.-S. Chua, Neural collaborative filtering, in: Proceedings of the 26th international conference on World Wide Web, 2017, pp. 173–182.
[8] X. Jiang, B. Hu, Y. Fang, C. Shi, Multiplex memory network for collaborative filtering, in: Proceedings of the 20th SIAM International Conference on Data Mining SIAM, 2020, pp. 91–99.
[9] H.-J. Xue, X. Dai, J. Zhang, S. Huang, J. Chen, Deep matrix factorization models for recommender systems, Proceedings of the 26th International Joint Conference on Artificial Intelligence, vol. 17, Melbourne, Australia, 2017, pp. 3203–3209.
[10] T. Ebesu, B. Shen, Y. Fang, Collaborative memory network for recommendation systems, in: Proceedings of the 41st international ACM SIGIR conference on Research and development in Information Retrieval, 2018, pp. 515–524.
[11] X. Wang, X. He, M. Wang, F. Feng, T.-S. Chua, Neural graph collaborative filtering, in: Proceedings of the 42nd international ACM SIGIR conference on Research and Development in Information Retrieval, 2019, pp. 165–174.
[12] R. v. d. Berg, T.N. Kipf, M. Welling, Graph convolutional matrix completion, arXiv preprint arXiv:1706.02263.
[13] 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.
[14] J. Wang, A.P. De Vries, M.J. Reinders, Unifying user-based and item-based collaborative filtering approaches by similarity fusion, in: Proceedings of the 29th international ACM SIGIR conference on Research and Development in Information Retrieval, 2006, pp. 501–508.
[15] J. Sun, Y. Zhang, C. Ma, M. Coates, H. Guo, R. Tang, X. He, Multi-graph convolution collaborative filtering, in: Proceedings of the 19th IEEE International Conference on Data Mining, IEEE, 2019, pp. 1306–1311.
[16] Y. Qu, T. Bai, W. Zhang, J. Nie, J. Tang, An end-to-end neighborhood-based interaction model for knowledge-enhanced recommendation, in: Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data, 2019, pp. 1–9.

[17] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, Line: Large-scale information network embedding, in: Proceedings of the 24th international conference on World Wide Web, 2015, pp. 1067–1077.

[18] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: online learning of social representations, in: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, 2014, pp. 701–710.

[19] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907.

[20] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, arXiv preprint arXiv:1710.10903.

[21] S. Zhang, L. Yao, A. Sun, Y. Tay, Deep learning based recommender system: a survey and new perspectives, ACM Comput. Surv. 52 (1) (2019) 1–38.

[22] I. Goodfellow, Y. Bengio, A. Courville, Y. Bengio, Deep Learning, vol. 1, MIT Press, Cambridge, 2016.

[23] M. Gao, J. Zhang, J. Yu, J. Li, J. Wen, Q. Xiong, Recommender systems based on generative adversarial networks: a problem-driven perspective, Inf. Sci. 546 (2020) 1166–1185.

[24] X. Guo, W. Lin, Y. Li, Z. Liu, L. Yang, S. Zhao, Z. Zhu, Dken: deep knowledge-enhanced network for recommender systems, Inf. Sci. 540 (2020) 263–277.

[25] J. Ni, Z. Huang, J. Cheng, S. Gao, An effective recommendation model based on deep representation learning, Inf. Sci. 542 (2021) 324–342.

[26] X. He, T.-S. Chua, Neural factorization machines for sparse predictive analytics, in: Proceedings of the 40th international ACM SIGIR conference on Research and Development in Information Retrieval, 2017, pp. 355–364.

[27] J. Han, L. Zheng, H. Huang, Y. Xu, S.Y. Philip, W. Zuo, Deep latent factor model with hierarchical similarity measure for recommender systems, Inf. Sci. 503 (2019) 521–532.

[28] D. Kim, C. Park, J. Oh, H. Yu, Deep hybrid recommender systems via exploiting document context and statistics of items, Inf. Sci. 417 (2017) 72–87.

[29] D. Hyun, C. Park, J. Cho, H. Yu, Learning to utilize auxiliary reviews for recommendation, Inf. Sci. 545 (2020) 595–607.

[30] E.R. Núñez-Valdéz, J.M.C. Lovelle, O.S. Martínez, V. García-Díaz, P.O. De Pablos, C.E.M. Marín, Implicit feedback techniques on recommender systems applied to electronic books, Comput. Hum. Behav. 28 (4) (2012) 1186–1193.

[31] G. Jawaheer, M. Szomszor, P. Kostkova, Comparison of implicit and explicit feedback from an online music recommendation service, in, in: Proceedings of the 1st international workshop on information heterogeneity and fusion in recommender systems, 2010, pp. 47–51.

[32] H.-C. Wang, H.-T. Jhou, Y.-S. Tsai, Adapting topic map and social influence to the personalized hybrid recommender system, Inf. Sci.

[33] L. Sheugh, S.H. Alizadeh, A novel 2d-graph clustering method based on trust and similarity measures to enhance accuracy and coverage in recommender systems, Inf. Sci. 432 (2018) 210–230.

[34] Z. Zhang, H. Lin, K. Liu, D. Wu, G. Zhang, J. Lu, A hybrid fuzzy-based personalized recommender system for telecom products/services, Inf. Sci. 235 (2013) 117–129.

[35] E.R. Núñez-Valdez, D. Quintana, R.G. Crespo, P. Isasi, E. Herrera-Viedma, A recommender system based on implicit feedback for selective dissemination of ebooks, Inf. Sci. 467 (2018) 87–98.

[36] H. Pei, B. Wei, K.C.-C. Chang, Y. Lei, B. Yang, Geom-gcn: geometric graph convolutional networks, arXiv preprint arXiv:2002.05287.

[37] L.F. Ribeiro, P.H. Saverese, D.R. Figueiredo, struc2vec: learning node representations from structural identity, in: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, 2017, pp. 385–394.

[38] J.L. Bentley, Multidimensional binary search trees used for associative searching, Commun. ACM 18 (9) (1975) 509–517.

[39] S.M. Omohundro, Five balltree construction algorithms, International Computer Science Institute Berkeley (1989).

[40] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, Bpr: Bayesian personalized ranking from implicit feedback, arXiv preprint arXiv:1205.2618.

[41] S. Rendle, Factorization machines, in: Proceedings of the 10th IEEE International Conference on Data Mining, IEEE, 2010, pp. 995–1000.

[42] C. Zhang, L. Yu, Y. Wang, C. Shah, X. Zhang, Collaborative user network embedding for social recommender systems, in: Proceedings of the 17th SIAM international conference on data mining SIAM, 2017, pp. 381–389.

[43] C. Feng, J. Liang, P. Song, Z. Wang, A fusion collaborative filtering method for sparse data in recommender systems, Inf. Sci. 521 (2020) 365–379.

[44] B. Yang, Y. Lei, J. Liu, W. Li, Social collaborative filtering by trust, IEEE Trans. Pattern Anal. Mach. Intell. 39 (8) (2016) 1633–1647.

[45] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the 13th international conference on artificial intelligence and statistics, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[46] J. Herce-Zelaya, C. Porcel, J. Bernabé-Moreno, A. Tejeda-Lorente, E. Herrera-Viedma, New technique to alleviate the cold start problem in recommender systems using information from social media and random decision forests, Inf. Sci. 536 (2020) 156–170.

[47] A. Li, B. Yang, Hsr: Hyperbolic social recommender, arXiv preprint arXiv:2102.09389.

[48] H. Wang, F. Zhang, J. Wang, M. Zhao, W. Li, X. Xie, M. Guo, Exploring high-order user preference on the knowledge graph for recommender systems, ACM Trans. Inf. Syst. 37 (3) (2019) 1–26.