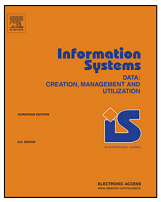
Information Systems 107 (2022) 102000

Contents lists available at ScienceDirect

Information Systems

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



Bilateral knowledge graph enhanced online course recommendation

Shuang Yang [a,*], Xuesong Cai [b]

[a] School of Software Engineering, East China Normal University, Shanghai, China
[b] Shanghai Synergy Digital Technology Innovation Institute, Shanghai, China

ARTICLE INFO

Article history:
Received 25 June 2021
Received in revised form 9 January 2022
Accepted 6 February 2022
Available online 9 February 2022
Recommended by Yannis Manolopoulos

Keywords:
Recommender system
Online course recommendation
Knowledge graph
Cold start
Personalized recommendation

ABSTRACT

Recommender system can provide users with items that meet their potential needs in mass information. Its development provides new ideas and supporting technologies for applications in online education scenarios. The previous recommendation methods usually only consider the enhancement of the item side, but ignore the importance of the user characteristics to the recommendation, and are not suitable for the online education scenario. To address this problem, we take knowledge graph as the auxiliary information source of collaborative filtering and propose an end-to-end framework using knowledge graph to enrich the semantics of the item representation. In particular, faced with the thorny problem of cold start, the framework makes use of the static features of users to personalize the modeling of new users. Experimenting with two public datasets and an industrial dataset, we demonstrate that the framework has significant performance improvements over the baseline and can maintain satisfactory performance with sparse user–item interactions.

1. Introduction

With the development of the Internet and the popularity of mobile terminals, the education mode gradually develops from offline to the combination of online and offline. Many education institutions begin to offer various online courses, such as Youdao classical courses, GET App, Huawei classical courses, and so on. With the explosive growth of the number of online courses, it is more and more difficult to accurately express the needs of users in the information flooding environment. Alleviating the influence of information overload and accurately helping users find courses to meet their personalized needs have become the main research goal of the recommender system.

One of the most representative methods in recommender systems is collaborative filtering (CF), which makes use of the user's historical interactions and recommends according to the common preferences of users [1,2]. However, the recommendation methods based on CF often have problems of cold start and sparse interaction data. To solve this problem, He et al. [3] propose a neural collaborative filtering (NCF) method based on CF, in which the matrix factorization (MF) method is transformed into a whole neural framework. In the NCF model, auxiliary data can be used to enhance the richness and accuracy of semantic representations of users or items [4–8]. However, the current NCF method only focuses on the feature of the item (e.g. commodities, movies, etc.), ignoring the important impact of the user's characteristics on the recommendation, which cannot meet the actual needs of the online course recommendation in the education scenario.

In recent years, knowledge graph (KG) has attracted more and more attention because it can mine the potential semantic association between entities, alleviate the sparsity problem, and improve the quality of recommendation [9–13]. The knowledge graph embedding (KGE) methods are applied to preprocess the knowledge graph, embedding the entities and relationships into the low-dimensional vector space, and preserving the structural information of the knowledge graph [14]. Subsequently, graph neural networks (GNNs) [15] and graph convolution networks (GCNs) [16] are introduced into the recommender system due to their good graphic representation learning ability.

Although some satisfactory results have been obtained, there are still some limitations. First, most traditional recommendation methods only model historical user–item interactions to capture user's implicit preferences, and ignore the significance of the static characteristics of users for revealing potential user preferences. Second, a few studies consider the item attributes but ignore the effects of different attribute relations on item modeling. For example, as shown in Fig. 1, the course "Deep Learning" is connected to the course "Machine Learning" by two kinds of attribute relations. They have the same instructor and belong to the same subject areas. Users who choose both courses may do so because they have the same "instructor" or because they are in the same "subject". This example shows that the attributes of an item are not equally important for each interaction, and should not be calculated simply on average.

* Corresponding author.
E-mail addresses: 52205902021@stu.ecnu.edu.cn, sh_cxs@163.com (S. Yang), caixuesong@ssdtii.cn (X. Cai).

https://doi.org/10.1016/j.is.2022.102000
0306-4379/

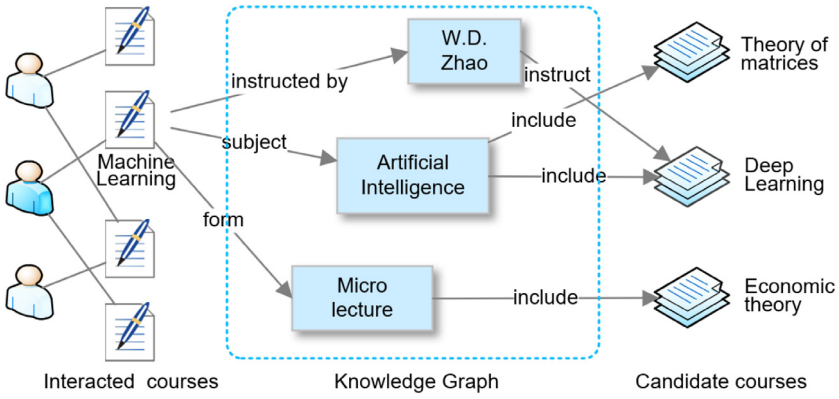


Fig. 1. An example of course recommendation with knowledge graph.

Third, there is little research on the user static characteristics in the existing works. In the actual recommendation process, although a user's preference for items comes from complex sources [17], it is closely related to his characteristics (such as age, industry, position, knowledge level, etc.). Making full use of this kind of information can provide useful auxiliary information for building a more accurate recommendation model, which is of great significance to solve the problem of cold start.

To address the limitations of previous works, we propose a **P**ersonalized **O**nline **C**ourse **R**ecommendation **(POCR)** framework. Firstly, we built an end-to-end framework and applied KG as an auxiliary information source. Secondly, KG-based propagation and aggregation techniques are applied to capture the potential features of the items for modeling. When aggregating neighbor nodes, an attention network is used to express the contribution of different kinds of attribute relations to the latent vector. Thirdly, we divide user modeling into two categories: static profile modeling and implicit preference modeling. For the new users who lack interactive information, the static profile modeling is done by building the user graph and capturing information about similar users. And for regular users, the implicit preference modeling is accomplished by historical interactions. Finally, the latent vectors of user and item are fed into a deep neural network (DNN) to predict the interaction probability.

To sum up, the main contributions of this paper are as follows:

- We propose an end-to-end recommendation framework based on improved CF, called the POCR framework, which uses KGs to enhance item-side and user-side information.
- We consider the importance of item attribute relationship types in revealing potential user preferences and introduce attribute-level attention mechanisms to capture the different contributions of attribute relationship types to item modeling.
- We describe user modeling into two categories. For the new users or the users with less interaction, the interactions of similar users are used to simulate the preferences of the new user, which effectively alleviates the problem of cold starts and sparse data.
- We conducted evaluation experiments on two public datasets and an industrial dataset, and the results confirm the effectiveness and superiority of the framework.

The structure of this paper is as follows. Section 2 introduces the literature related to the research. Section 3 is the problem statement. Section 4 presents the proposed framework. Section 5 describes the experimental results and discussion. Section 6 provides the conclusion.

2. Related work

Recommendation technology has shown some obvious development trends in recent years. Among various methods, CF-based methods [18–22] are one of the most widely used models in recommender systems. CF provides personalized item recommendation for users by learning user and item embedding from users' historical interactions [1,23]. The CF-based algorithms most widely involved in early studies are BPR [24], matrix factorization (MF) [25], etc. The traditional MF method directly obtains the prediction score through vector inner product, which limits the expressibility of the MF method [1].

With the rise of deep learning, it has become a focus of research to use deep learning to learn the nonlinear interaction between users and items, triggering another development peak of recommendation technology. Many recommendation models based on deep learning techniques have been proposed, such as Wide&Deep [26], DMF [27], NeuACF [28]. He Xiangnan proposed NCF [1], which combines linear matrix factorization and nonlinear deep neural network to model the implicit user–item interactions, and has a great improvement in recommendation performance compared with the traditional MF method [1]. However, traditional NCF does not consider auxiliary information, such as pictures, texts, social networks, and other meaningful data, and cannot make full use of a variety of information for fusion recommendation, so there are certain limitations in performance. To solve this problem, many methods based on deep learning try to integrate auxiliary information for the recommendation, among which knowledge graph is an important branch.

Early research on KGs focuses on path-based models, which leverage various connection modes of entities to recommend, such as RKGE [29], Hete-MF, and HeteRec [30]. The disadvantage of path-based models is that they rely heavily on hand-designed paths. Most of the subsequent studies are embedding-based methods, which encodes KG as low-rank embeddings and uses the embeddings to make further recommendations, such as CKE [31], SHINE [32], and DKN [33]. Embedding-based methods are more flexible but do not focus on the information connectivity pattern in the KG. The hybrid approaches based on the above two types combine the advantages of both, such as RippleNet [34], RippleNet-agg [35], and IntentGC [36], etc.

In recent years, GCNs have attracted more and more attention due to their performance in processing graphic structure data [37, 38] and have been proven to be one of the best-performing structures for various graphic learning tasks [39–41]. Researchers have developed many recommendation models based on GCNs. For example, Ying et al. [42] develop a GCN network, which combines random walk and graph convolution technology to obtain node

embedding for the web-scale recommendation. Wu et al. [43] propose a graph neural network recommendation method based on the session (SRGNN), using GCN for the sequence prediction task. Kipf et al. [44] introduce a variant of GCN that can be extended to large graphs and use a semi-supervised approach for node representation learning. Song et al. [45] use the social graph to spread information between users and propose an attention-based social network (DGRec). Fan et al. [46] propose a GNN model (Graphrec), which combines the user social graph and user–item graph to complete the recommendation. Hamilton et al. [47] propose a method, which samples neighbor nodes and aggregates their features to generate embedding based on the embedding method of GCNs.

In terms of model training, many KG-based recommender system models usually adopt a multi-stage algorithm process, including a model for extracting graph features and a model for predicting the link, which are separately trained. However, literature [48,49] shows that graph structure data using an end-to-end modeling learning approach can effectively improve link prediction results because it can fully share the convolution weights and auxiliary information is allowed to be introduced in the form of node features.

In addition, CF-based methods are usually limited by cold start and data sparsity. For a new user or the user with less interaction, it is difficult to recommend him accurately and effectively due to the lack of interactive data to mine potential preferences. Many efforts have also been made to deal with these problems. Some researchers build cross-domain recommender systems [50,51], which use rich information of the source domain to improve prediction performance in the target domain with sparsity. Others use the social network as a medium to obtain users' potential preferences [52,53]. These approaches are more challenging when applied to online learning scenarios, as it is difficult to find cross-domain users with similar behaviors to online learning platforms, and it is not easy to build social networks for all platform users. However, inspired by these approaches, we have new ideas about the definition and application of similar users in the field of education.

In this paper, we study the experiences and practices of previous researches and propose the POCR model, which considers the importance of different attribute relationships of items to users, and adopts an attention mechanism to capture users' preferences more accurately. To solve the problem of cold start and data sparsity, we propose a strategy to simulate new user preferences by using the historical preferences of similar users and use static information representing users' inherent attributes as the main basis for computing similarity.

3. Problem statement

In this article's recommendation scenario, the inputs include three parts: user–item interaction matrix, user similarity matrix, and item graph. For the interaction part, $\mathcal{U} = \{u_1, u_2, \ldots, u_M\}$ represents users and $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ represents items, and M and N are numbers of users and items, respectively. The user–item interaction matrix is represented as $Y = \{y_{uv} | u \in \mathcal{U}, v \in \mathcal{V}\}$, which is constructed based on the implicit feedback of users as follows:

$$y_{uv} = \begin{cases} 1 & \textit{if interaction } (u, v) \textit{ is observed}; \\ 0 & \textit{otherwise}. \end{cases}$$

If there is an interaction between user u and item v (e.g. user u clicks item v), then $y_{uv} = 1$, otherwise $y_{uv} = 0$. The item graph $\mathcal{G}$ consists of entity-relation-entity triples (h, r, t), where h, t, and r respectively represent the head node, the tail node, and the relationship between the head and tail node. The recommendation problem we need to solve can be attributed to calculating the probability that the user u interacts with item v which has not interacted before. The equation is $\hat{y}_{uv} = \mathcal{F}(u, v|\theta, Y, \mathcal{G})$, where θ is the parameter of function $\mathcal{F}$.

4. The POCR framework

In this section, we present our framework. Firstly, we briefly describe the overall framework. Then, we introduce each component of the framework in detail. Finally, we describe prediction and loss function.

4.1. Architecture of the POCR framework

The POCR framework consists of three main parts (Fig. 2): the KGE module, user modeling module, and recommendation module. In the KGE module, entity embedding propagation and aggregation are completed on KG, and the entity embedding part of the item representation is obtained. Depending on the type of user, user modeling is divided into two categories, historical interaction modeling, and static information modeling. In the recommendation module, POCR uses an attention network to generate embedding of user implicit preferences for item interaction and takes the user latent embedding and the item latent embedding as inputs to output the predicted probability.

4.2. KG-enhanced item modeling

In the recommendation module, the embedding of each item consists of two parts: its generic representation z_i and its entity embedding representation q_i. The entity embedding representation is obtained by capturing the item features associated with it in the KG. The final embedding representation v_i of the item is obtained by adding the two parts: $v_i = z_i + q_i$.

4.2.1. Entity embedding representation of item

The modeling of the item entity embedding part is completed in the KGE module. We construct a heterogeneous knowledge graph for items and take items and their attributes as nodes in the graph. This graph is used to complete multi-attribute aware item entity embedding modeling and obtain the item latent embedding. The schematic diagram of the item graph is shown in Fig. 3.

We adopt a GCN-based method for entity embedding. The representation and propagation process of entity embedding includes two basic operations: propagation and aggregation. In the propagation process, the features of nodes propagate iteratively to the neighbor nodes along the connection between nodes. In the process of aggregation, the features of neighbor nodes are aggregated to get the embedding of target nodes. Each convolution operation means that the aggregation of local neighbors in the previous layer generates the embedding of the current node.

In the actual situation, different users have different sensitivity to different item attributes. When interacting with items, users tend to give priority to the attributes with high sensitivity, while the attributes with low sensitivity will be considered later. For example, when selecting an online course, some users regard "subject" as the first factor of their choice, while some users consider "instructor" as the first factor. From this, we know that different relationship types that connect entities are of different importance when modeling items. Therefore, based on the basic aggregation method, we add an attention mechanism that can express the weight of different relationship types. The entity embedding process that is aware of the weights of attribute relationship types is described in detail in the next subsection.

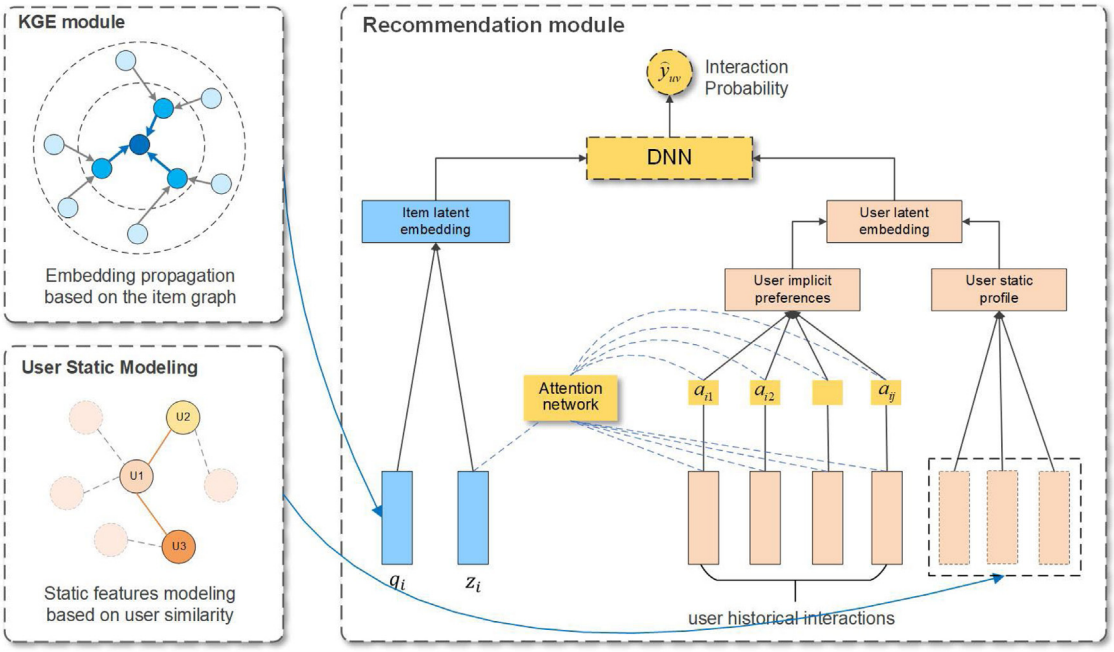


Fig. 2. The architecture of the proposed POCR framework.

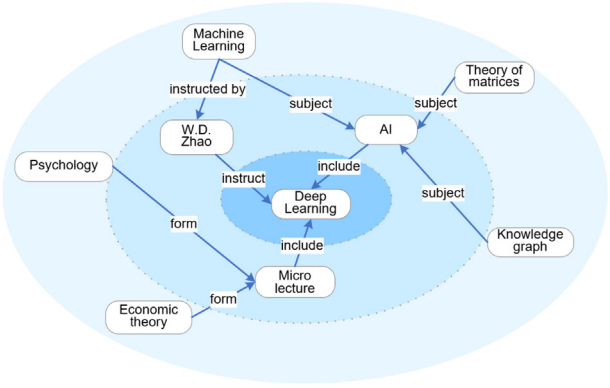


Fig. 3. Illustration of the item graph with key attributes.

4.2.2. Multi-attribute aware neighbor aggregation

To reflect the importance of attribute relationship types in the aggregation into latent entity embedding, we treat the neighbors of different relationship types differently and assign a weight value β_r for each relationship type to represent its importance in the aggregation. The specific process is as follows.

First, neighbors with the same relationship type are aggregated.

$$t_r = \sum_{j \in N_i(r)} \frac{1}{C_{i,r}} W_1^{(l)} q_j^{(l)} \tag{1}$$

where t_r represents the average of the neighbor vectors of relation r and $W_1^{(l)}$ is the weight matrix. $N_i(r)$ is the set of neighbors of the target node in relation r and $C_{i,r} = |N_i(r)|$ represents the number of neighbors of the target node in relation r. Then calculate the weight of each relationship. When solving the weights of each relation type, we design a two-layer attention network, which sets different weights for different relation types.

$$\beta_r^* = W_2^T \cdot \sigma \left(W_1 \cdot \left[t_r \bigoplus q_i^{(l)} \right] + b_1 \right) + b_2 \tag{2}$$

$$\beta_r = \frac{exp(\beta_r^*)}{\sum_{r \in N(r)} exp(\beta_r^*)} \tag{3}$$

where β_r^* is the weight of relation type r. $\bigoplus$ is a concatenation operator and b_1 and b_2 are the bias. β_r is the normalized representation of β_r^*. $q_i^{(l)} \in R^d$ is the latent embedding of the target node q_i in the l-layer network, σ is an activation function. Finally, the neighbors with different relationship types are then aggregated and then aggregated with the target node.

$$q_i^{(l+1)} = \sigma \left(\sum_{r \in R} \beta_r t_r + W_0^{(l)} q_i^{(l)} + b_0 \right) \tag{4}$$

where $q_i^{(l+1)} \in R^d$ is the latent embedding of node q_i in the $l+1$ layer network.

For each node in KG, the KGE module encodes its neighborhood information into real-valued features in an unsupervised way. By doing this, we can not only get the local structure of the nodes but also capture the high-order structural proximity between nodes in the graph through propagation. After propagation and aggregation, the entity embedded representation of the item is obtained, which is fused with its generic representation, and then sent to the recommendation module for prediction.

Algorithm 1: Entity embedding in KGE module

Input: Knowledge graph $\mathcal{G} = (\mathcal{E}, \mathcal{R})$; depth L; weight matrices W^l, $\forall l \in \{1, \ldots, L\}$; non-linearity σ
Output: Entity embedding q_i;
1 **for** $l = 1 \ldots L$ **do**
2 **for** $e_i \in \mathcal{E}$ **do**
3 **for** $r \in N_i(r)$ **do**
4 Calculate the aggregated neighbors t_r according to Eq.(1);
5 Calculate the weights β_r according to Eq.(2) and (3);
6 **end for**
7 Calculate the latent embedding $q_i^{(l+1)}$ according to Eq.(4);
8 **end for**
9 **end for**
10 **return** q_i (i.e. $q_i^{(L)}$)

The algorithm of the entity embedding in the KGE module is shown in Algorithm 1. For a given number of layers l (line 1),

for each entity e_i (line 2), we first calculate the average representation vector t_r of its neighbors under relation r (line 4). And then we calculate the weight factor β_r (line 5), then aggregate its neighbors with its representation $q_i^{(l)}$ to obtain the one to be used at the next iteration (line 7). After L iterations, the final q_i is the output vector $q_i^{(L)}$(line 10).

4.2.3. Entity embedding training

We learn the entity embedding representation of the item in an unsupervised way. We put the pairs of positive and negative entities into the loss function, and try to make the positive entities closer to the target entity and the negative entities farther away from the target entity in the low-dimensional vector space. Stochastic gradient descent is used to optimize the parameters.

$$\mathcal{J}_{\mathcal{G}}(z_u) = -log(\sigma(z_u^T z_v)) - Q \cdot E_{v_n \sim P_n(v)} log(\sigma(-z_u^T z_v)) \tag{5}$$

where v denotes the positive entity of the target entity u within fixed range, v_n denotes the negative entity. P_n represents the sampling distribution and Q denotes the number of negative samples. σ is the sigmoid function.

4.3. User modeling by static profile and historical interaction

In the traditional CF method, the user preference can be obtained by mining the user's historical interactions, and then his response to the target item can be predicted. However, this could be difficult for a new user without any interaction or the ones with less interaction. Inspired by the idea of the CF method which assumes that similar users have similar preferences, we can predict the preference of a new user for the target item by finding users who are similar to him and using their preferences to describe the preferences of the new user.

4.3.1. Similarity-based user static feature modeling

The first task of similarity-based new user modeling is to find users who are similar to the target users and select the best-predicted users for sampling from a large number of similar users.

Many platforms require users to fill in some necessary information when registering, such as age, industry, position, and so on. From this information, the platform can form an initial profile of each user. Such information changes little over time and is collectively referred to as the static features of the user.

Intuitively, two people who share more of the same static features are more similar. This is especially true in the education scenario, where users who are similar in age, occupation, and knowledge level tend to show higher similarity when choosing courses. For example, a younger user tends to choose the necessary certification courses or basic introductory professional courses, while older users tend to choose courses that are in-depth in a technical field or management courses. In addition, users in the same industry and similar positions tend to take more similar courses.

To make it easier to find similar users, we build a heterogeneous user graph. With users and their static attributes as nodes in the graph, the constructed graph is shown in Fig. 4(a). Two users with common attributes are similar, and similar users are connected through different relationship types, as shown in Fig. 4(b).

For simplicity, we assume that there is no substantial difference in the expression of node similarity between different relationship types and that the number of identical attributes reflects the degree of similarity. Two user entities are directly connected, and the number of common attributes between users is taken as the weight of the connecting edge, which further simplifies the graph into the weighted undirected homogeneity graph shown in Fig. 4(c). The user graph data is stored in the form of a user adjacency matrix and a user similarity matrix for use, as shown in Fig. 4(d). Then, based on user similarity matrix, the processes of neighbor sampling and item sampling are performed.

Considering that the number of neighbors of each user varies greatly, we adopt the fixed-size neighbor sampling method in the model. If neighbors are randomly selected, information of important neighborhood nodes is likely to be lost. Therefore, we choose relatively important neighbor nodes as sampling targets. By calculating the similarity, a list of nodes sorted by similarity can be obtained. The first K_u neighbors in the list are selected as the sampled neighborhood by the maximum sampling method. When the number of neighbors is less than K_u, select all neighbors as the neighborhood and randomly choose neighbors to supplement the neighborhood. All sampled neighbors form a subset $s_a^n(u)$. For each sampled neighbor, K_v items are selected from his historical interactions. All sampled items form the sampled items subset $s_a^i(u)$. By aggregating the features of all sampled items, the preference of the target user can be approximated.

Use a toy example to illustrate the main process of sampling historical interactions of similar users as shown in Fig. 5. The new user u_1 is a programmer who has just graduated from college and works in education industry. By analyzing his static features, we find his similar users u_2, u_3 and u_4, with similarities of 4, 3 and 4 respectively. Assume that the neighborhood sample number K_u is set to 3 and the item sample number K_v is set to 2. According to the ranking of neighbor similarity, u_2 and u_4 are selected for sampling at first, and 2 items are respectively selected from their interaction items. After all the neighbors with similarity of 4 (u_2 and u_4) are sampled, the number of sampled neighbors is still less than 3, user u_3 with similarity of 3 is sampled. Finally, all 6 sampled items from 3 similar users are sampled, and the sampled items are put into $s_a^i(u)$, which will be used to simulate the preferences of u_1.

4.3.2. User implicit preference modeling with history interaction

User implicit preference modeling is obtained by the historical interaction between the user and the items. The latent embedding based on user history interaction is as follows:

$$p_u = \sigma\left(W \cdot Agg_u\left(\left\{h_j, \forall j \in C(u)\right\}\right) + b\right) \tag{6}$$

where $C(u)$ is the set of items that have interacted with user u, h_j is the embedding vector of these items, σ is the nonlinear activation function, b and W are the bias and weight matrix, respectively. Agg_u is the aggregation function, the most commonly used is the mean operator:

$$p_u = \sigma\left(W \cdot \left\{\sum_{j \in C(u)} \frac{1}{C(u)} h_j\right\} + b\right) \tag{7}$$

In practice, however, user preferences may be multifaceted. The contribution of each historical interaction to the judgment of the target item may be completely different. The mean-value method does not accurately describe the different contributions of each interaction to the prediction. So, we use an attention network to calculate the contribution of different historical interactions in modeling user implicit preferences:

$$p_u = \sigma\left(W \cdot \left\{\sum_{j \in C(u)} a_{ij} h_j\right\} + b\right) \tag{8}$$

where a_{ij} is the attention weight of interaction h_j when constructing user u's preference from historical interaction $C(u)$. The attention equation is as follows:

$$a_{ij}^* = {W_2}^T \cdot \sigma\left(W_1 \cdot \left[z_i \odot h_j\right] + b_1\right) + b_2 \tag{9}$$

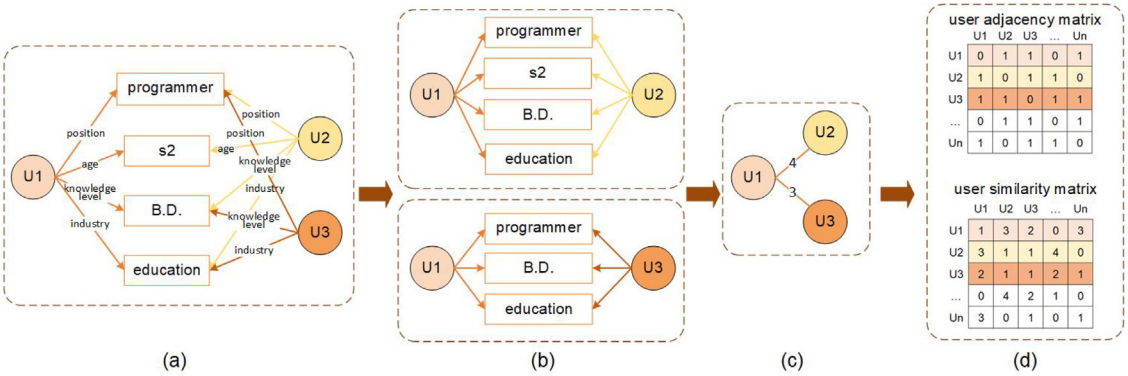


Fig. 4. User graph construction and simplification.

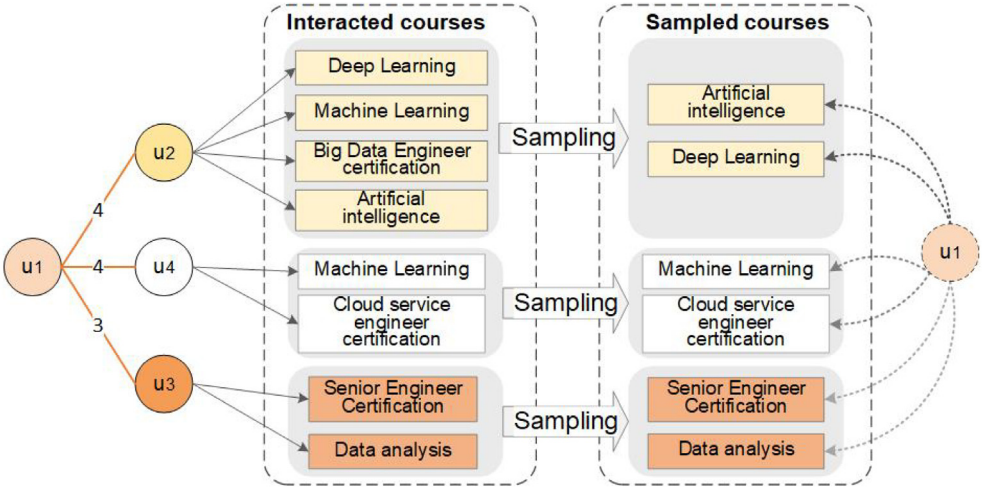


Fig. 5. Simulating user preference by sampling historical interactions of similar users.

where z_i is the candidate item, W_1 is the weight matrix, which transforms the inputs to the hidden layer, and vector ${W_2}^T$ transforms the hidden layer to the attention weight. $\odot$ is the inner product operator. The larger the attention weight, the greater the contribution of the historical interaction to the user preference representation.

To make the weight of different interactions easier to compare, the attention score is normalized:

$$a_{ij} = \frac{exp(a_{ij}^*)}{\sum_{j \in C(u)} exp(a_{ij}^*)} \tag{10}$$

The normalized attention score is used to calculate the weighted sum of all historical interactions as the implicit preference of the user.

4.4. Model prediction and loss function

In the prediction part, the traditional method is to take the user embedding and item embedding as two independent inputs and predict them through inner product or simple concatenation [54]. However, simple inner product or vector concatenation is not enough to describe the potential correlation between user and item in the education scenario. To solve this problem, we send the concatenated vector into a three-layer DNN to predict:

$$\hat{y}_{uv} = DNN(p_u, v_i) \tag{11}$$

where p_u and v_i are the final embedding vector of user u and item v. We choose the *ReLU* function as the activation function of DNN. In this sense, we can give the model a high level of flexibility and nonlinear modeling capabilities. The complete loss function of our model is as follows:

$$\min \mathcal{L} = \sum_{(u,v) \in D} \mathcal{I}(y_{uv}, \hat{y}_{uv}) + \alpha[-log(\sigma(z_u^T z_v)) - Q \cdot \mathbb{E}_{v_n \sim P_n(v)} log(\sigma(-z_u^T z_v))] \tag{12}$$

where $\hat{y}_{uv}$ is the predicted value, indicating the probability of user u interacting with item v.

Algorithm 2: Learning algorithm for POCR

Input: user-item interaction matrix Y; Knowledge graph $\mathcal{G}$; sampled neighborhood $S_a^n(u)$; parameter:Θ
Output: Prediction function $\mathcal{F}$ (u, v |Θ, Y, $\mathcal{G}$)

```
While POCR does not converge do
    for (u, v) in Y do
        If u is a new user do
            Get the sampled item set S_a^i(u) from S_a^n(u);
            Calculate user's latent embedding p_u according to Eq.(8) - (10),
            where replace C(u) with S_a^i(u) in Eq.(8) and (10);
        else do
            Calculate user's latent embedding p_u according to Eq.(8) - (10);
        end else
        Calculate item entity embedding q_i by Algorithm 1;
        v_i = z_i + q_i;
        Calculate interaction probability ŷ_uv according to Eq.(11);
        Update parameters by gradient descent;
    end for
end while
return F
```

Table 1
Basic statistics and hyperparameter settings for the three datasets (K: neighbor sampling size, d: dimension of embeddings, L: maximum propagation depth).

Dataset	#users	#items	#interactions	#KG triples	Hyper-parameters
Movielens-20M	116,148	13,453	10,671,395	393,782	$K = 4$, $d = 32$, $L = 1$
Book-Crossing	17,460	14,530	135,648	105,816	$K = 8$, $d = 32$, $L = 2$
OCP dataset	9,986	2,378	103,162	8,387	$K = 4$, $d = 64$, $L = 2$

The complete algorithm is shown in Algorithm 2. For a given user–item pair $(u, (v)$ (line 2), if user u is a new user, the sampled item set $S_a^i(u)$ is first obtained by sampled neighborhood $S_a^n(u)$ (line 4), and then simulated user preference is obtained by calculating the sampled items in $S_a^i(u)$ (line 5); Otherwise, the user's preferences are mined from their historical interactions (line 7). Following the steps described in Algorithm 1, we can obtain a latent embedding vector of the candidate item v_i (line 9,10) which is fed into a DNN together with user representation p_u for predicting the probability (line 11,12).

5. Experiments

5.1. Datasets

To evaluate the effectiveness of our method, we conduct experiments on two public datasets and an industrial dataset.

Movielens-20M[1] This dataset includes 20,000,263 ratings (ranging from 1 to 5) applied to 27,278 movies by 138,493 users.

Book-Crossing[2] This dataset includes 139,746 ratings (ranging from 0 to 10) of books in the Book-Crossing community.

Industrial dataset: We collected the data on Huawei Online Course Platform[3] and sampled 103,162 click records of 9,986 users on 2,378 courses. We call this industrial dataset the Online Course Platform (OCP) dataset for short. To verify the performance of the POCR model on a cold start, the test set includes new users who have not participated in any courses.

For Movielens-20M and book-Crossing datasets, we need to convert explicit feedback to implicit feedback. Each implicit feedback is considered a positive rating and is marked as 1, and negative examples are randomly selected from items that have never interacted. According to the characteristics of the two datasets, we set the positive rating threshold of MovieLens-20M to 4, and all ratings in Book-Crossing to positive sample due to their sparsity [13,50]. Directly applying the code provided by authors of [13,50], we built the knowledge graphs for the two datasets from Microsoft Satori.[4]

The static features of the user in the OCP dataset are formed by user id, age, industry, position, knowledge level, etc. Historical interaction information includes user id, course id, and click count (0 means no click, 1 means click). Key attributes such as subject, instructor, industry, technical direction, and course form are selected when constructing the course graph. The basic statistics of the three datasets are presented in Table 1.

5.2. Model comparison

We compare the POCR framework with the following methods:

•NCF [1] is a neural network architecture for collaborative filtering. It uses neural structure instead of the inner product to learn an arbitrary matching function from data.

•CKE [28] is a recommendation method based on embedding, which mines semantic embedding of items from structural information, text, and visual content to assist CF prediction. In model comparison, we only use structural information as an assistant for CKE.

•DeepWide [55] is a general deep model for recommendation, combining a (wide) linear channel with a (deep) non-linear channel. We use the embeddings of users and items as inputs to feed both channels.

•RippleNet [13] is an approach based on knowledge graph embedding that uses graph propagation and aggregation techniques to capture users' potential preferences.

•KGCN [56] uses GCN technology to capture neighborhood information in KG to obtain the correlation between items. It uses an attention mechanism when aggregating neighborhoods, and the recommendation results are personalized.

•KCRec [57] is an end-to-end framework that aggregates item features by propagating neighbor relationships in KG to get a representation of user preferences. The user adjacency graph is constructed by using the similarity features of different users to obtain the high-order representation of users.

•KGR [48] is a KG-based collaborative filtering method. The embedding of the user and the item are based on the attributes of items in KG and aggregated with the general representation of users and items by MF for an implicit recommendation.

In addition, to test the effectiveness of KG as auxiliary information and the impact of attention mechanism in the framework, we design several simplified variants of POCR and compare them with the complete framework. The experimental results of the variants are shown in Sections 5.4.1 and 5.4.2.

5.3. Experiment setup

In the experiment, the ratio of the training set, validation set, and test set is 6:2:2. The three parts of the OCP dataset are divided according to the chronological order of interaction data, with the oldest interaction as the training set and the latest interaction as the test set. The test set results are used as experimental results. Our approach is evaluated in a CTR scenario. The positive rating threshold is set to 0.5, that is to say, if $\hat{y}_{uv} \geq 0.5$, the item will be recommended to the user. The trained model is applied to the test set and the predicted click probability is output. We use AUC and F1 values as the evaluation criterion for CTR prediction. Each experiment is repeated five times, and the average results of the five experiments are recorded.

The hyperparameters are given in Table 1. The settings of hyperparameters are determined by optimizing AUC on the validation set. For knowledge propagation modeling in KG, the last layer of the framework uses the *tanh* function, and the other layers use *ReLU* function.

5.4. Results

In this section, we describe the results of the experiments.

1 https://grouplens.org/datasets/movielens/.
2 http://www2.informatik.uni-freiburg.de/~cziegler/BX/.
3 https://e.huawei.com/cn/talent/#/.
4 https://searchengineland.com/library/bing/bing-satori.

Table 2
Comparison of different models.

Models	Movielens-20M		Book-Crossing		OCP dataset	
	AUC	F1	AUC	F1	AUC	F1
NCF	0.906	0.825	0.637	0.594	0.641	0.592
CKE	0.891	0.801	0.619	0.574	0.630	0.583
DeepWide	0.922	0.843	0.682	0.603	0.735	0.667
RippleNet	0.925	0.847	0.691	0.617	0.743	0.672
KGCN	0.932	0.854	0.702	0.632	0.756	0.688
KCRec	0.948	0.871	0.731	0.661	0.765	0.694
KGR	0.950	0.872	0.743	0.673	0.773	0.703
POCR	–	–	–	–	**0.797**	**0.733**
POCR-C	**0.958**	**0.881**	**0.755**	**0.687**	0.776	0.706
POCR-N	0.907	0.827	0.640	0.601	0.648	0.595

Table 3
AUC with different ratios of the training set on MovieLens-20M.

Models	Ratios				
	20%	40%	60%	80%	100%
NCF	0.710	0.791	0.832	0.877	0.906
CKE	0.723	0.784	0.816	0.850	0.891
DeepWide	0.741	0.804	0.841	0.892	0.922
RippleNet	0.753	0.817	0.866	0.894	0.925
KGCN	0.767	0.839	0.875	0.914	0.932
KCRec	0.853	0.873	0.886	0.924	0.948
KGR	0.858	0.875	0.888	0.923	0.950
POCR-C	0.890	0.919	0.933	0.947	0.958

5.4.1. Comparison of different models

The comparison results of different models are shown in Table 2, from which we have the following observations:

- Compared with CF-based methods with auxiliary information, the methods without any KG information (e.g. NCF) are relatively less effective.
- RippleNet is a structure-based method, which applies a breadth-first search strategy to capture multi-hop neighbor information about an entity. But RippleNet also showed some limitations in capturing the complicated interactions between users and items.
- KGCN mines users' potential preferences by aggregating the information of multi-hop neighbors, to learn the representation of entities. It achieves good results on MovieLens-20M dataset with more entity relationships, and worse on Book-crossing and OCP dataset with smaller scale knowledge graphs.
- KCRec and KGR are two newly proposed models. KCRec is an end-to-end framework, and KGR is a KG-based collaborative filtering method that aggregates the general representation of users and items to make an implicit recommendation by the MF method. These two methods perform well in both rich and sparse datasets.
- POCR-C and POCR-N in Table 2 are two variants of the POCR model, which will be described in 5.4.2. POCR-C removes the user's static modeling module. Compared with the KGR model with the best performance in Table 2, the AUC of the POCR-C model on Movielens-20M increases by 0.8%, and the F1 increases by 0.9%. The AUC and F1on Book-Crossing increase by 1.2% and 1.4% respectively. In terms of the OCP dataset, POCR performed best, with AUC increasing by 2.4% and F1 increasing by 3.0%. This shows that the POCR model is effective and can make a more accurate recommendation for users by successfully utilizing information from both sides. POCR-N removes the functions of KG assistance and user static modeling, and the model degenerates into an NCF model without any assistance, resulting in serious performance degradation.

5.4.2. Comparison among POCR variants

We design several variants for comparative experiments. The variant POCR-C removes the user's static modeling module from the complete framework, which is used to evaluate the effect of the model on the cold start problem. The variant POCR-N removes KG-based item modeling and user static modeling. The experimental results of these two variants are shown in Table 2.

Another variant is built by changing the attention network within the framework. The variants are POCR-A_{I} with attention in item modeling removed, POCR-A_{C} with attention in user history interaction modeling removed, and POCR-A_{N} with both two attention networks removed.

We test the performance of the variants on the OCP dataset, and the results are shown in Fig. 6. The attention network applied in the item graph increased by 2.2% and 1.0% in AUC and F1, respectively. The attention network applied in historical interactive information increased by 3.7% and 1.7% in AUC and F1, respectively. The combined use of the two attention networks, AUC and F1 of the whole framework increased by 4.8% and 2.8% respectively, indicating the effectiveness of the introduction of two attention networks in the framework.

5.4.3. Performance in sparse scenarios

To study the performance of the POCR model in the case of data sparsity, we use Movielens-20M dataset for verification and analysis. We reduced the amount of data in the training set by 20% each time while keeping the validation set and test set unchanged.

As shown in Table 3, the random sampling rate decreases from 100% to 20%. With the reduction of the training samples, the performance of all baseline methods deteriorates. When the number of training sets was reduced to 20% of the complete training set, the AUC of the baseline model decreased by 21.6%, 18.9%, 19.6%, 18.6%, 17.7%, 10.0% and 9.7%, respectively. By contrast, POCR-C's AUC dropped by only 7.1%, indicating that the POCR-C framework can maintain good performance even when user-course interactions are sparse.

5.5. Parameter sensitivity

5.5.1. The impact of parameter α

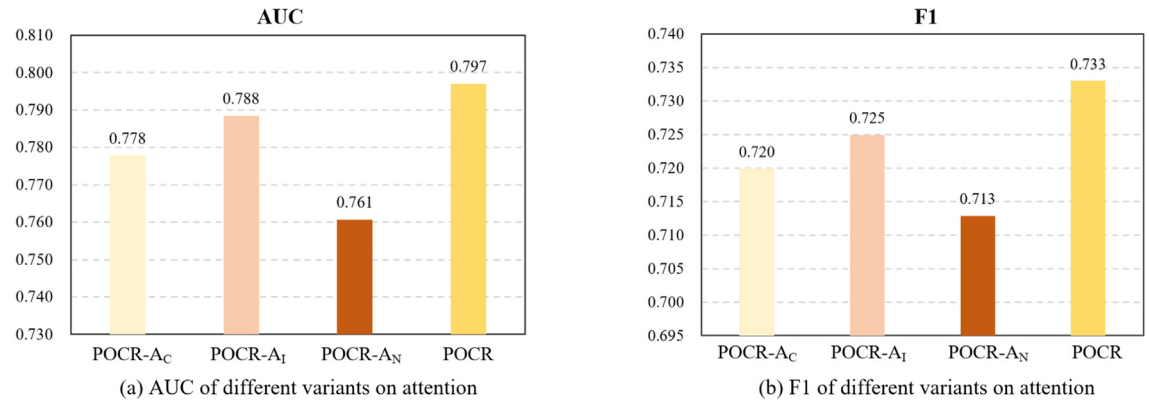
In the complete loss function expression shown in Eq. (12), parameter α determines the tradeoff between KG embedding and prediction. The greater the value of α, the greater the weight of knowledge graph embedding loss. As α changes, changes in AUC can be seen from Fig. 7.

With the increase of α, the AUC index shows obvious improvement at the beginning, but when the value of α increases to 0.17, the AUC index begins to decline. In an extreme case, when $\alpha = 0$, the POCR model degenerates into the NCR without any auxiliary information of KG, and the model depends heavily on the user–item interactions for performance. As α increases, the impact of KG embedding in item modeling increases. When the value of α exceeds the equilibrium point, the loss function is too inclined to the embedding learning of KG, resulting in the performance decline. Therefore, we set the value of α to 0.17.

5.5.2. The effect of layer number in KG

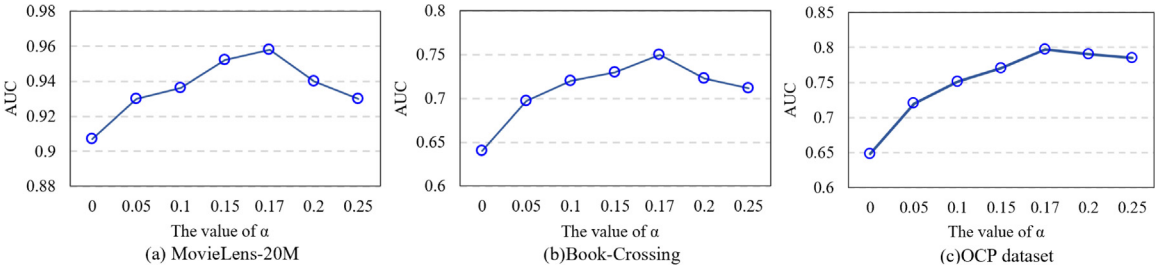
By changing the maximum propagation layer L, we adjust the local KG context neighborhood to observe the performance changes.

We observe the overall performance of the framework by changing the maximum number of layers of the context neighborhood of the knowledge graph. For Book-crossing and OCP datasets, it can be seen from Table 4 that with the increase of the number of layers, there is no higher AUC as expected. On the contrary, starting from $L = 2$, AUC decreases with the increase



(a) AUC of different variants on attention (b) F1 of different variants on attention

Fig. 6. Comparison results among POCR variants on OCP dataset.



(a) MovieLens-20M (b)Book-Crossing (c)OCP dataset

Fig. 7. Impact of parameter α on three datasets.

Table 4
The impact of embedding propagation layers.

#layer	MovieLens-20M		Book-Crossing		OCP dataset	
	AUC	F1	AUC	F1	AUC	F1
$L = 1$	**0.958**	**0.881**	0.729	0.658	0.763	0.714
$L = 2$	0.950	0.872	**0.755**	**0.687**	**0.797**	**0.733**
$L = 3$	0.931	0.853	0.747	0.678	0.784	0.721
$L = 4$	0.910	0.830	0.702	0.631	0.728	0.660

of the number of layers. For MovieLen-20 m dataset, the best performance occurs at $L = 1$.

The possible reason is that the nodes in the first layer and the second layer are the fields that can best express the characteristics of the target node. They not only reflect the similarity between nodes but also preserve the difference between nodes, so they can better describe the potential characteristics of the target nodes. For the smaller graph, the larger the number of layers, the more likely to overlap. For the larger graph, the number of nodes in the neighborhood will increase exponentially when the number of layers increases by one, so noise may be introduced to reduce the accuracy. The propagation layers of each dataset are set according to the L value with the best practical effect.

6. Conclusions and future work

We propose POCR, an end-to-end KG-enhanced collaborative filtering framework, for online course recommendation in this paper. POCR applies knowledge graph information to provide rich and fresh auxiliary information for recommendation and distinguishes the importance of different attribute relationship types in item modeling. The model can not only capture rich semantic information but also capture the hidden relationship between users and recommended items. Compared with other similar methods, the other advantage of POCR is that it uses the preferences of similar users to simulate the preference of the new user, which provides a new idea for the preference mining of new users and effectively solves the problem of cold start.

In the research of this paper, the user static modeling based on similarity does not consider the correlation between key attributes, such as age and knowledge level. In subsequent studies, we will consider the correlation between attributes to further optimize the model. In addition, we will try to change the sampling number and ranking method of similar users and their historical interactions, and continue to research ensuring recommendation accuracy and considering recommendation diversity.

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.

References

[1] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.-S. Chua, Neural collaborative filtering, in: Proc. 26th Int. Conf. World Wide Web, 2017, pp. 173–182.
[2] L. Chen, L. Wu, R. Hong, K. Zhang, M. Wang, Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach, Proc. AAAI Conf. Artif. Intell. 34 (1) (2020) 27–34.
[3] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.-S. Chua, Neural collaborative filtering, in: international conference on world wide web, 2017, pp. 173–182.
[4] G. Hu, Y. Zhang, Q. Yang, CoNet: Collaborative cross networks for cross-domain recommendation, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, 2018, pp. 667–676.

[5] L. Luo, H. Xie, Y. Rao, F.L. Wang, Personalized recommendation by matrix co-factorization with tags and time information, Expert Syst. Appl. 119 (2019) 311–321.
[6] S. Qian, T. Zhang, C. Xu, Cross-domain collaborative learning via discriminative nonparametric Bayesian model, IEEE Trans. Multimed. 20 (8) (2018) 2086–2099.
[7] P. Symeonidis, D. Malakoudis, Multi-modal matrix factorization with side information for recommending massive open online courses, Expert Syst. Appl. 118 (2019) 261–271.
[8] E. Zheng, G.Y. Kondo, S. Zilora, Q. Yu, Tag-aware dynamic music recommendation, Expert Syst. Appl. 106 (2018) 244–251.
[9] Y. Cao, X. Wang, X. He, Z. Hu, T.-S. Chua, Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences, in: The World Wide Web Conference, 2019, pp. 151–161.
[10] V. Sourabh, C. Chowdary, Peer recommendation in dynamic attributed graphs, Expert Syst. Appl. 120 (2019) 335–345.
[11] H. Wang, M. Zhao, X. Xie, W. Li, M. Guo, Knowledge graph convolutional networks for recommender systems, in: The World Wide Web Conference, 2019, pp. 3307–3313.
[12] H. Wang, F. Zhang, M. Zhao, W. Li, X. Xie, M. Guo, Multi-task feature learning for knowledge graph enhanced recommendation, in: WWW '19: The World Wide Web Conference, 2019, 2019, pp. 2000–2010, http://dx.doi.org/10.1145/3308558.3313411.
[13] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, Minyi Guo, RippleNet: PRopagating user preferences on the knowledge graph for recommender systems, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, ACM, 2018.
[14] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, Minyi Guo, Graphgan: Graph representation learning with generative adversarial nets, in: AAAI, 2018, pp. 2508–2515.
[15] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Trans. Neural Netw. 20 (1) (2009) 61–80.
[16] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: Advances in Neural Information Processing Systems, 2016, pp. 3844–3852.
[17] A. Ahmed, K. Saleem, U. Rashid, A. Baz, Modeling trust-aware recommendations with temporal dynamics in social networks, IEEE Access 8 (2020) 149676–149705.
[18] Zhiyong Cheng, Xiaojun Chang, Lei Zhu, Rose C. Kanjirathinkal, Mohan Kankanhalli, MMALFM: EXplainable recommendation by leveraging reviews and images, TOIS 37 (2) (2019) 16.
[19] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, YongDong Zhang, Meng Wang, LightGCN: SImplifying and powering graph convolution network for recommendation, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2020, pp. 639–648.
[20] Fan Liu, Zhiyong Cheng, Changchang Sun, Yinglong Wang, Liqiang Nie, Mohan Kankanhalli, User diverse preference modeling by multimodal attentive metric learning, in: Proceedings of the 27th ACM International Conference on Multimedia, ACM, 2019, pp. 1526–1534.
[21] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, Tat-Seng Chua, KGAT: KNowledge graph attention network for recommendation, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2019, pp. 950–958.
[22] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, Tat-Seng Chua, Neural graph collaborative filtering, in: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2019, pp. 165–174.
[23] X. Wang, X. He, M. Wang, F. Feng, T.-S. Chua, Neural graph collaborative filtering, in: Proc. 42nd Int. ACM SIGIR Conf. Res. Develop.Inf. Retr. Jul. 2019, pp. 165–174.
[24] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, Lars Schmidt-Thieme, BPR: Bayesian personalized ranking from implicit feedback, in: UAI, 2009, pp. 452-461.
[25] Yehuda Koren, Robert Bell, Chris Volinsky, Matrix factorization techniques for recommender systems, IEEE Comput. 42 (2009) 42–49.
[26] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al., Wide & deep learning for recommender systems, in: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, ACM, 2016, pp. 7–10.
[27] H.-J. Xue, X. Dai, J. Zhang, S. Huang, J. Chen, Deep matrix factorization models for recommender systems, in: Proc. 26th Int. Joint Conf. Artif. Intell. Aug. 2017, pp. 3203–3209.
[28] C. Shi, X. Han, S. Li, X. Wang, S. Wang, J. Du, P. Yu, Deep collaborative filtering with multi-aspect information in heterogeneous networks, IEEE Trans. Knowl. Data Eng. 17 (2019) Early Access, http://dx.doi.org/10.1109/TKDE.2019.2941938.
[29] Z. Sun, J. Yang, J. Zhang, A. Bozzon, L. Huang, C. Xu, Recurrent knowledge graph embedding for effective recommendation, in: Proceedings of the 12th ACM Conference on Recommender Systems, 2018, pp. 297–305.
[30] X. Yu, X. Ren, Y. Sun, B. Sturt, J. Han, Recommendation in heterogeneous information networks with implicit user feedback, in: Proceedings of the 7th ACM Conference on Recommender Systems, 2013, pp. 347–350.
[31] F. Zhang, N.J. Yuan, D. Lian, X. Xie, W.Y. Ma, Collaborative knowledge base embedding for recommender systems, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 353–362.
[32] H. Wang, F. Zhang, M. Hou, X. Xie, M. Guo, Q. Liu, SHINE: Signed heterogeneous information network embedding for sentiment link prediction, in: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, 2017, pp. 592–600.
[33] H. Wang, F. Zhang, X. Xie, M. Guo, Dkn: Deep knowledge aware network for news recommendation, in: Proceedings of the 2018 World Wide Web Conference, 2018, pp. 1835–1844.
[34] H. Wang, F. Zhang, J. Wang, M. Zhao, M. Guo, RippleNet: Propagating user preferences on the knowledge graph for recommender systems, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, 2018, pp. 417–426.
[35] 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. (TOIS) 37 (3) (2019) 1–26.
[36] J. Zhao, Z. Zhou, Z. Guan, W. Zhao, W. Ning, G. Qiu, X. He, IntentGC: a scalable graph convolution framework fusing heterogeneous information for recommendation, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 2347–2357.
[37] Thomas N. Kipf, Max Welling, Semi-supervised classification with graph convolutional networks, in: Proc. Int. Conf. Learning Representations, 2017.
[38] Ziqi Liu, Chaochao Chen, Longfei Li, Jun Zhou, Xiaolong Li, Le Song, Yuan Qi, 2019. Geniepath: Graph neural networks with adaptive receptive paths, in: Proc. AAAI Conf. Artificial Intelligence.
[39] Keyulu Xu, Weihua Hu, Jure Leskovec, Stefanie Jegelka, How powerful are graph neural networks? in: Proc. Int. Conf. Learning Representations (ICLR), 2019, URL.
[40] Yingxue Zhang, Soumyasundar Pal, Mark Coates, Deniz Üstebay, Bayesian graph convolutional neural networks for semi-supervised classification, in: Proc. AAAI Int. Conf. Artificial Intelligence, 2019.
[41] Tyler Derr, Yao Ma, Jiliang Tang, Signed graph convolutional networks, in:2018 IEEE International Conference on Data Mining (ICDM), IEEE, 2018, pp. 929–934.
[42] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, Jure Leskovec, Graph convolutional neural networks for web-scale recommender systems, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ACM, 2018, pp. 974–983.
[43] Wu Shu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, Tieniu Tan, Session-based recommendation with graph neural networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 346–353.
[44] Thomas N. Kipf, Max Welling, Semi-supervised classification with graph convolutional networks, in: International Conference on Learning Representations (ICLR), 2017.
[45] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, Jian Tang, Session-based social recommendation via dynamic graph attention networks, in: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, ACM, 2019, pp. 555–563.
[46] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, Dawei Yin, Graph neural networks for social recommendation, in: The World Wide Web Conference, ACM, 2019, pp. 417–426.
[47] Will Hamilton, Zhitao Ying, Jure Leskovec, Inductive representation learning on large graphs, in: Advances in Neural Information Processing Systems, 2017, pp. 1024–1034.
[48] Ya Chen, Samuel Mensah, Fei Ma, Hao Wang, Zhongan Jiang, Collaborative filtering grounded on knowledge graphs, Pattern Recognit. Lett. 151 (2021) 55–61, http://dx.doi.org/10.1016/j.patrec.2021.07.022.
[49] F. Tian, B. Gao, Q. Cui, et al., Learning deep representations for graph clustering, in: Proceedings of the National Conference on Artificial Intelligence, vol. 2, 2014, pp. 1293–1299.
[50] R. Wang, Z. Xie, G. Qi, P. Li, NAUI :Neural attentive user interest model for cross-domain CTR prediction, J. Phys. Conf. Ser. 1873 (1) (2021) 2021 2nd International Workshop on Electronic communication and Artificial Intelligence, IWECAI 2021; http://dx.doi.org/10.1088/1742-6596/1873/1/012091.
[51] Z. Wang, W. Xiao, et al., LHRM: AN LBS based heterogeneous relations model for user cold start recommendation in online travel platform, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics, in: LNCS, vol. 12534, 2020, pp. 479–490, Neural Information Processing - 27th International Conference, http://dx.doi.org/10.1007/978-3-030-63836-8_40.

[52] R. Wang, M. Gao, et al., JUST-BPR: Identify implicit friends with jump and stay for social recommendation, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), in: LNCS, vol. 12534, 2020, pp. 455–466, Neural Information Processing - 27th International Conference, ICONIP 2020, Proceedings, http://dx.doi.org/10.1007/978-3-030-63836-8_38.

[53] Feng Zou, Debao Chen, Qingzheng Xu, Ziqi Jiang, Jiahui Kang, A two-stage personalized recommendation based on multi-objective teaching–learning-based optimization with decomposition, Neurocomputing 452 (2021) 716–727, http://dx.doi.org/10.1016/j.neucom.2020.08.080.

[54] N. Srivastava, R.R. Salakhutdinov, Multimodal learning with deep boltzmann machines, in: NIPS, 2012, pp. 2222–2230.

[55] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al., Wide & deep learning for recommender systems, in: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, ACM, 2016, pp. 7–10.

[56] H. Wang, M. Zhao, X. Xie, W. Li, M. Guo, Knowledge graph convolutional networks for recommender systems, in: The World Wide Web Conference, 2019, pp. 3307–3313.

[57] Lisa Zhang, Zhe Kang, Xiaoxin Sun, Hong Sun, Bangzuo Zhang, Dongbing Pu, KCRec: KNowledge-aware representation graph convolutional network for recommendation, Knowl.-Based Syst. 230 (2021) 107399, http://dx.doi.org/10.1016/j.knosys.2021.107399.