Expert Systems With Applications 213 (2023) 118984

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Exploring indirect entity relations for knowledge graph enhanced recommender system

Zhonghai He [a,1], Bei Hui [a,1], Shengming Zhang [a], Chunjing Xiao [b,*], Ting Zhong [a,*], Fan Zhou [a]

[a] School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China
[b] Henan Key Laboratory of Big Data Analysis and Processing, Henan University, Kaifeng 475004, China

ARTICLE INFO

Keywords:
Recommender systems
Knowledge graph
Graph neural networks
Exposure bias
Data sparsity

ABSTRACT

Knowledge graph (KG)-based recommendation models generally explore auxiliary information to alleviate the sparsity and cold-start problems in recommender systems. Previous approaches enhance representations of users and items by exploring the influence of multi-hop neighbors. However, existing works fail to consider the indirect feedback for improving user representation and the diversity of the multi-hop neighbors for enriching item representation. To this end, we present a novel recommender system, called **E**ntity **R**elation **S**imilarity and **I**ndirect **F**eedback-based **K**nowledge graph enhanced **R**ecommendation (ERSIF-KR) to enhance representation learning in KG-based recommender systems. In addition, our model exploits indirect feedback of items that are not directly interacted with users to alleviate the exposure bias while enhancing user similarity computation when learning user representation. Moreover, our method directly incorporates representation of multi-hop neighbors into the target item embedding with weights determined by the correlations between high-order and low-order relations, which can significantly boost the item representation learning. Extensive experiments on three real-world datasets demonstrate that our model achieves remarkable gains in terms of recommendation performance and model convergence time, and effectively alleviates the sparsity and cold start problems.

1. Introduction

Recommender systems have been widely deployed in many online services, including social networks, search engines, news websites, and E-commerce platforms. Among recommendation strategies, knowledge graph (KG) – where nodes and edges denote the entities and relations, respectively – is an ideal tool for depicting the side information and diverse relations associated with users/items (Palumbo et al., 2020; Yang & Dong, 2020). Recently, numerous studies have adopted KGs to enhance the recommendation performance (Guo et al., 2020; Wang, Li et al., 2021; Wang, Zhao et al., 2019), aiming to address various problems such as data sparsity and cold-start users confronted by collaborative filtering (CF) techniques and deep neural networks models (Guo et al., 2022; Son, 2016; Xiao et al., 2020). Previous KG-based recommendation methods depend on manual feature engineering and require extensive computation on entity/relation representation learning. To capture the rich interactions and auxiliary information in KG, graph neural networks (GNNs) (Wu, Pan et al., 2021; Zhong et al., 2020) that aggregate node attributes from neighborhoods using neural networks are usually exploited to learn the representations in KGs in an efficient, explicit, and end-to-end manner (Wang, Zhao et al., 2019; Ying et al., 2018). In this way, auxiliary information can be exploited to enrich the representations of users, entities, and relations, such as high-order interactions (Wang, He, Cao et al., 2019; Wang, He, Wang et al., 2019; Wang, Zhang, Zhao et al., 2019), user-specific relations (Cao et al., 2019; Wang, Zhang, Zhang et al., 2019; Zhou et al., 2020), as well as structural proximity (Lu et al., 2018; Sang et al., 2021).

Challenges: Despite their effectiveness and promising performance, several crucial factors are missing in existing KG-based recommender systems. First, previous studies mainly exploit the user–item links when searching similar neighbors to enhance user embeddings. While the un-interacted items are not fully explored for users' embedding. Some works even simply regard un-interacted items as negative (disliked) samples. However, it may raise *bias* issue, e.g., most of these items are just not viewed because they are not exposed to users instead of dislike (Chen et al., 2019), which is a.k.a. *exposure bias* (Chen, Dong, Wang et al., 2021). Therefore, the un-interacted items should not be entirely regarded as negative samples, and some of them can be taken into account for seeking similar users to enhance user representation.

* Corresponding authors.
E-mail addresses: hzh@uestc.edu.cn (Z. He), bhui@uestc.edu.cn (B. Hui), shmizhang@gmail.com (S. Zhang), chunjingxiao@gmail.com (C. Xiao), zhongting@uestc.edu.cn (T. Zhong), fan.zhou@uestc.edu.cn (F. Zhou).
[1] Contribute equally to this work.

https://doi.org/10.1016/j.eswa.2022.118984
Received 14 March 2022; Received in revised form 4 October 2022; Accepted 4 October 2022
Available online 12 October 2022

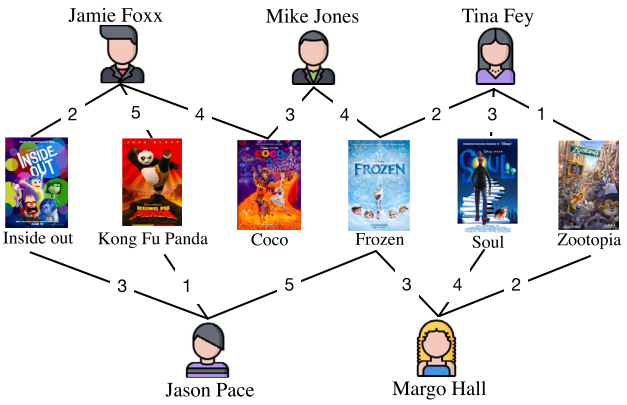


Fig. 1. An example of computing user similarity by considering indirect feedback. *Frozen*, which is not rated by Jamie Foxx, should also be considered to compute the similarity degree of Jamie Foxx and Jason Pace because the close relationship between *Coco* and *Frozen* indicates that both users share similar interest on the topic related to both movies.

Fig. 2. The knowledge graph related to the target movie *Edge of Tomorrow*. When embedding the movie *Edge of Tomorrow*, the two-hop neighbor *Twilight* should play a more important role than *Titanic*, since the first-order relation r_4 (Genre1_Fantasy) is more similar to the second-order relation r_5 (Genre2_Science_fiction) than to another second-order relation (r_3: *Director*).

An example of the user-movie interaction graph is shown in Fig. 1, where we would like to compute the similarity of user Jamie Foxx and Jason Pace for downstream representation learning and movie recommendation. In addition to the common interested movies *Inside out* and *Kong Fu Panda*, *Frozen* that has not been rated by Jamie Foxx may also contribute to the similarity computation of both users, because *Frozen* and *Coco* have very high similarity, suggesting the common interest of the two users over the two movies. Consequently, exploiting indirect interactions and mitigating exposure bias may potentially benefit similar user aggregation to improve user representation learning.

Second, few approaches explicitly consider the correlations between different-order relations when aggregating multi-hop neighbors. Existing KG-based models iteratively incorporate the high-order node embedding into the low-order one and adopt the closest neighbor embedding to improve item representation (Wang, He, Cao et al., 2019; Wang, He, Wang et al., 2019). Nevertheless, they fail to consider the correlations between the high-order relations and low-order relations, which can reflect the importance of multi-hop neighbors in enhancing the target item representation. Because the types of items that users have interacted with are usually very broad, the high-order neighbors might have completely different meanings with the latent semantic representation of the target item (Abu-El-Haija et al., 2019; Schlichtkrull et al., 2018), which would introduce noise into embeddings when hierarchically aggregating neighbors without considering their correlations. Therefore, to efficiently aggregate neighbors, their correlations should be considered, i.e., a two-hop neighbor of the target item should play a more important role in computing the target item embedding if its one-order relation and two-order relation have a higher similarity. For example, as shown in Fig. 2, compared with the two-hop neighbor (*Titanic*), *Twilight* should contribute more to the computation of the target movie (*Edge of Tomorrow*) embedding, because its high-order relation r_5 (*Fantasy*) and low-order relation r_4 (*Science fiction*) are more similar than r_3 and r_4. In other words, the similarity of different-order relations should be adopted to determine multi-hop neighbors' importance for enhancing item representation.

Our Contributions: To address the aforementioned issues, we present a novel KG-based recommender system ERSIF-KR (*Entity Relation Similarity and Indirect Feedback-based Knowledge graph enhanced Recommendation*). Specifically, It consists of an indirect feedback-based user representation learning network for enhancing user embedding. This module considers the indirect feedbacks (unobserved items) and assigns each item an attention score to dynamically adjust their weight in computing the similarity between users. The similar users are then incorporated into the layer-wise propagation rule of the graph neural networks to optimize users' embeddings. To enrich entity representation, we design another relation similarity-based item network, which adopts the similarity degree of high-order relations and low-order relations to determine the importance of multi-hop neighbors on the computation of entity representation. Specifically, we present a KG embedding method based on the RotatE (Sun et al., 2019) to transform relations into feature vectors, which is subsequently used to compute relation similarity degree. Besides, during the optimization process of this KG embedding, we introduce the Radial Basis Function (RBF) as the distance measure to improve the representations. As a result, the representations of users, items, as well as neighbors are fed into the prediction network to calculate the users' clicking probability for the item. Overall, our contributions are summarized as follows:

- We present a novel KG-based recommender system that considers users' indirect feedback and importance differences between items to construct the user similarity network. It not only allows us to incorporate more users' preferences into user representation learning but also alleviates the exposure bias of previous models that are purely dependent on interacted items.
- Instead of treating different relations between items in KG independently, we consider the similarity degree of different-order relations to identify the importance of multi-hop neighbors in enhancing item representation.
- To effectively learn the embeddings of entities and relations, we introduce a distance function to enhance the rotation-based KG embedding model. In addition, we present three types of aggregators to merge embeddings of the neighbors.
- We conduct extensive experiments on real-world datasets. The comprehensive evaluations show that, compared to the state-of-the-art baselines, our model achieves 2.8%–3.2% and 5%–15% improvements in terms of prediction accuracy and convergence time, respectively.

The rest of this paper is organized as follows. Section 2 reviews the literature related to exposure biases and recommender systems. Section 3 describes the problem formulation and necessary backgrounds. Section 4 presents the details of the our ERSIF-KR model. Section 5 compares our model with baseline approaches on three real-world datasets. Finally, we conclude this work with future directions in Section 6.

2. Related work

In this section, we review the relevant literature from three aspects, including exposure biases in recommender systems, high-order connectivity aggregation, and KG-based recommendation, and position our work in the context.

2.1. Exposure biases in recommender systems

Exposure bias is raised because users are only exposed to a part of specific items, while a large number of unobserved interactions are not recorded but cannot represent negative preference (Chen, Dong, Wang et al., 2021). Due to numerous items, the unobserved feedback might be attributed to unknown or unexposed items rather than disliked ones. Hence, unobserved interactions do not always represent negative preference. Exposure bias is very common in recommender systems and without carefully handling the bias issues, the effectiveness of the recommender systems is dubious in online development and user satisfaction (Chen, Dong, Qiu et al., 2021; Saito et al., 2020).

To alleviate exposure bias, a strategy is to perform sampling by designing different sampling strategies, which determine which data are used to update parameters and how often, and thus scale the data contribution (Chen, Jiang et al., 2021; Ding et al., 2019; Wang et al., 2020b). Another solution is to assign different confidence weights to unobserved interactions based on item popularity and user preference level since popular items and interesting items are more likely to be exposed (Chen, Wang et al., 2020; Saito et al., 2020; Wang, Chen et al., 2021). Recently, confounder analysis in causal inference is adopted to solve the bias in recommendation systems (Li et al., 2022; Wang, Feng, et al., 2021).

Similar to these works, our proposed module, indirect feedback-based user profiling, selects some unobserved items (samples) for optimizing user embeddings, which can be regarded as a kind of method of performing sampling for alleviating exposure bias. While different from these existing works, our module assigns each item an attention score to dynamically adjust their weights in computing the similarity between users. The similar users are then incorporated into the layer-wise propagation rule of the graph neural networks to optimize user embeddings.

2.2. High-order connectivity aggregation for recommendation

Exploiting high-order connectivity is of importance to perform high-quality recommendation. Based on the homophily principle (Zhu et al., 2020), high-order connections (multi-hop neighbors) are generally aggregated using GNNs to enhance item embedding, which has been successively applied in recommender systems (Guo & Wang, 2021; Wu, Sun et al., 2021; Yin et al., 2019). Existing works usually incorporate multi-hop neighbors by hierarchical neighbor aggregation, i.e., each layer only aggregates one-hop neighbors, and multiple layers are introduced to incorporate multi-hop neighbors. They try to design various aggregators to collect and aggregate user, item, and neighborhood information to improve the performance of recommender systems, such as the summation or element-wise product of two representations (Wang, He, Cao et al., 2019; Wang, He, Wang et al., 2019; Wang, Zhao et al., 2019). Since the types of items that users have interacted with are usually very broad, the high-order interactions might have completely different meanings with the latent semantic representation of the target item. Hence, the neural attention mechanism has been utilized to learn different weights of neighbors in KG when they are in different conditions during neighbor aggregation (Sang et al., 2021; Wang, He, Cao et al., 2019; Wang et al., 2020a).

However, the hierarchical neighbor aggregation splits a complete multi-hop neighbor into several one-hop neighbors, which might destroy the completeness of multi-hop neighbors. For example, for a multi-hop neighbor, if its two-hop relation is quite different from its one-hop relation, it should play a less important role in computing the target item embedding. However, the hierarchical neighbor aggregation fails to consider the connections of different-order relations, and might not properly discriminate the importance of multi-hop neighbors during neighbor aggregation. Hence, instead of adopting the hierarchical neighbor aggregation, our designed method, relation similarity-based item embedding, directly incorporates multi-hop neighbors into the target item embedding by considering correlations between high-order and low-order relations, which can effectively discriminate the importance of multi-hop neighbors during aggregation.

2.3. KG-based recommendation

Recently, improving recommendation performance by integrating the side information using the knowledge graph has attracted great interest. Existing KG-based recommendation models can be categorized into three types: path-based methods, embedding-based methods, and hybrid methods (Guo et al., 2022; Wang, Zhang, Zhang et al., 2019). Path-based models (Hu et al., 2018; Wang et al., 2018; Yu et al., 2014) discover various patterns (paths) of connections among entities and distinguish the importance between paths. For example, PER (Yu et al., 2014) considers different types of paths between users and items and learns personalized models for individuals to distinguish different relationships. MCRec (Hu et al., 2018) conducts a further study on meta-path based on PER. It leverages a sampling method to select high-quality path instances for building the meta-path based context and designs a three-way neural interaction model via explicitly incorporating meta-path based context. RippleNet (Wang et al., 2018) extends and propagates the user's implicit interests along with links in the knowledge graph.

Embedding-based methods (Ai et al., 2018; Zhang et al., 2016) apply various knowledge graph embedding (KGE) methods to facilitate the downstream recommendation. KGE, which aims to represent entities and relations as low dimensional vectors, is wildly applied in the task of predicting missing links. For example, TransE (Bordes et al., 2013) tries to conduct link prediction by embedding entities and relationships of multi-relational data in low-dimensional vector spaces. TransR (Lin et al., 2015), a variant of TransE, learns embeddings by first projecting entities from entity space to corresponding relation space and then building translations between projected entities. Motivated by Euler's identity: $e^{i\theta} = \cos\theta + i\sin\theta$, recent work RotatE (Sun et al., 2019) defines each relation as a rotation angle (θ) from the source entity to the target entity. Specifically, CFKG (Ai et al., 2018) embeds heterogeneous entities on the knowledge base networks. Additionally, CKE (Zhang et al., 2016) utilizes the TransR method to learn items' structural representations considering the heterogeneity of both nodes and relationships.

Hybrid models (Wang, He, Cao et al., 2019; Wang, Zhao et al., 2019) usually try to discover high-order connectivities between entities in an end-to-end manner. For example, knowledge graph attention network (KGAT) (Wang, He, Cao et al., 2019) explicitly models the high-order connectivities in KG in an end-to-end fashion. Meanwhile, it recursively propagates the embeddings from a node's neighbors to refine its embedding through an attention mechanism. Furthermore, KGNN-LS (Wang, Zhang, Zhang et al., 2019) discovers high-order structures and semantic information inherent in the KG and identifies the important relationships for a given user through the label smoothness regularization. Recently, attention techniques have been applied in exploiting the high-order connectivities for KG-based recommendation. For instance, HAGRec (Yang & Dong, 2020) proposes a hierarchical attention mechanism to adaptively characterize collaborative signals and explore users' potential preferences from the high-order connectivity structure of KG. CKAN (Wang et al., 2020a) develops a knowledge-aware attention mechanism to identify the contribution of different knowledge-based neighbors, which is further used to combine collaborative signals with knowledge associations together. In addition, some

other works (Chen, Zhang et al., 2020; Wang et al., 2020b) focus on designing new optimization algorithms from different perspectives, e.g., for better knowledge graph embedding and sampling (Chen, Zhang et al., 2020; Wang et al., 2020b).

The aforementioned KG-based recommendation models exploit auxiliary information to enhance recommendation performance. While different from them, we try to explore indirect feedback to alleviate exposure bias for optimizing user representation, and leverage the correlations between high-order and low-order relations to weight multi-hop neighbors for optimizing item embedding.

3. Preliminaries

Before introducing the proposed approach, we first define the knowledge graph-based recommendation problem and present the necessary backgrounds.

3.1. Knowledge graph-based recommendation

In a general recommendation scenario, we have a set of M users $\mathcal{U} = \{u_1, \dots, u_M\}$ and a set of N_t items $\mathcal{E} = \{i_1, \dots, i_{N_t}\}$. Let $\mathbf{Y} \in \mathbb{R}^{M \times N_t}$ be the user–item interaction matrix denoting users' indirect feedback, where $y_{u,i} = 1$ indicates that user u has clicked (or rated) item i; otherwise $y_{u,i} = 0$. $\mathbf{T} \in \mathbb{R}^{M \times N_t}$ indicating users' explicit feedback, where $t_{u,i} \in \mathbf{T}$ is a rating value that directly reflects user's preference over the rated item if user u has rated item i; otherwise $t_{u,i} = 0$. We denote historical interaction data as a user–item interaction graph, which is presented as $\{(u, y_{u,i}, i) | u \in \mathcal{U}, i \in \mathcal{E}\}$.

In addition to these interactions, relations between items are also taken into account in the knowledge graph. Formally, knowledge graph is set as $\mathcal{G} = \{(h, r, t) | h, t \in \mathcal{E}, r \in \mathcal{R}\}$, where each triple represents entity-relation-entity indicating a relation r between head the entity h and the tail entity t. $\mathcal{E} = \{i_1, \dots, i_{N_e}\}$ and $\mathcal{R} = \{r_1, \dots, r_L\}$ denote the set of N_e entities and L relations, respectively. In a typical item (e.g., movie, music or restaurant) recommendation scenario, an item $i \in \mathcal{E}$ corresponds to an entity (the head entity h or tail entity t). Note that, the notations used throughout the paper are described in Table 1.

3.2. Knowledge graph embedding using RotateE

RotatE model (Sun et al., 2019) vectorize relations and entities while preserving the knowledge graph structure. We try to learn the representations of relations between the head entity and tail entity based on the rotation-based KGE method (Sun et al., 2019; Zhang et al., 2020) for the entities with one-hop and two-hop neighbor connectivities. For each triplet (h, r, t), the RotatE model learns the embeddings of each entity and relation by optimizing the rotation principle: $\mathbf{v}_h \circ \mathbf{v}_r \approx \mathbf{v}_t$, where $\circ$ denotes the element-wise product. For the relations between entities with neighbor connectivity, we expect that:

$$\mathbf{v}_n = \mathbf{v}_i \circ \mathbf{v}_r, \tag{1}$$

$$\mathbf{v}_{n^{(2)}} = \mathbf{v}_n \circ \mathbf{v}_{r^{(2)}}, \tag{2}$$

where $\mathbf{v}_r \in \mathbb{R}^D$ and $\mathbf{v}_{r^{(2)}} \in \mathbb{R}^D$ are the embeddings of the first-order and second-order relations, $\mathbf{v}_i \in \mathbb{R}^D$, $\mathbf{v}_n \in \mathbb{R}^D$, and $\mathbf{v}_{n^{(2)}} \in \mathbb{R}^D$ are the embeddings of the tail entities.

3.3. Problem formulation

We take user–item interaction matrix $\mathbf{Y}$, user–item rating matrix $\mathbf{T}$, and knowledge graph $\mathcal{G}$ as our model's input to predict whether user u might be interested in item i without clicking (or rating) history. Our purpose is to leverage a prediction function to recommend a list of items to each user. The prediction function is defined as follows:

$$\hat{y}_{u,i} = \mathcal{P}(u, i | \mathbf{Y}, \mathbf{T}, \mathcal{G}, \Theta), \tag{3}$$

Table 1
List of notations.

Notation	Description
$\mathbf{W}_{(*)}$	The weight matrix
$\mathbf{Y}$	User-item interaction matrix
$\mathbf{P}$	Laplacian matrix for user clicking similarity
$\mathcal{U}/\mathcal{E}/\mathcal{R}$	Set of users/items/relations
$M/N_t/N_e/L$	The number of users/items/entities/relations
$\mathbf{v}_u/\mathbf{v}_i/\mathbf{v}_r$	User/item/relation embedding vector
$\mathcal{H}/\mathcal{T}$	Set of user rating records/similar neighbors
K	The dimension of hidden layers in user embedding optimization
Z	The neighbor sampling size
D	The latent dimension of embedding vector

where $\hat{y}_{u,i}$ indicates the predicted clicking probability of user u for item i, and Θ denotes the parameters of model function $\mathcal{P}$.

Utilizing these graphs for recommendation is rather challenging due to its high-dimension and heterogeneity properties. Therefore, users, items, and relations are generally represented as low dimensional vectors using knowledge graph embedding and graph neural networks according to the knowledge graph and the interaction graph. Hence, we denote $\mathbf{v}_u \in \mathbb{R}^D$, $\mathbf{v}_r \in \mathbb{R}^D$, and $\mathbf{v}_i \in \mathbb{R}^D$ as the embedding vectors of user u, relation r, and item (entity) i, respectively. Here D is the latent dimension of the embedding vector, and the initial embedding vectors of users, relations, and items are defined as follows:

$$\mathbf{V}_{(u)} = \left[\mathbf{v}_{u_1}, \dots, \mathbf{v}_{u_M}\right]^\top, \tag{4}$$

$$\mathbf{V}_{(i)} = \left[\mathbf{v}_{i_1}, \dots, \mathbf{v}_{i_{N_t}}\right]^\top, \tag{5}$$

$$\mathbf{V}_{(r)} = \left[\mathbf{v}_{r_1}, \dots, \mathbf{v}_{r_L}\right]^\top, \tag{6}$$

$$\mathbf{V}_{(n)} = \left[\mathbf{v}_{n_1}, \dots, \mathbf{v}_{n_{N_e}}\right]^\top, \tag{7}$$

where $\mathbf{V}_{(u)} \in \mathbb{R}^{M \times D}$, $\mathbf{V}_{(i)} \in \mathbb{R}^{N_t \times D}$, $\mathbf{V}_{(r)} \in \mathbb{R}^{L \times D}$, and $\mathbf{V}_{(n)} \in \mathbb{R}^{N_e \times D}$ are four embedding look-up tables optimized in an end-to-end manner. Additionally, the embedding $\mathbf{v}_i$ corresponds to a head entity vector $\mathbf{v}_h$. Hence, taking these embedding vectors as input, the prediction function becomes:

$$\hat{y}_{u,i} = \mathcal{P}(u, i | V_{(u)}, V_{(i)}, V_{(r)}, V_{(n)}, \Theta). \tag{8}$$

To enrich the representations of users, relations, and items, we propose a novel model ERSIF-KR to leverage underlying characteristics and patterns in the user–item interaction graph and knowledge graph.

4. Model description

This section presents the details of the ERSIF-KR model. The overall framework is illustrated in Fig. 3, which consists of three main components: (1) Indirect feedback-based user module that exploits indirect interactions and essential difference of items to filter similar users for refining user representation; (2) Relation similarity-based item module that directly incorporates multi-hop neighbors into the target item embedding with weights based on the correlations of the high-order and low-order relations; (3) The prediction layer that aggregates the optimized user vectors, the preference of similar users, the item embeddings, as well as the neighbor entity influence to generate the user's clicking probability for a specific item.

4.1. Indirect feedback-based user profiling

In personalized recommendation, user preferences can be discovered by aggregating the behavior from similar users (Ye et al., 2011; Zhong et al., 2020), which is a.k.a. collaborative filtering. Therefore, considering the target user's similar neighbors may enrich the representation. In general, existing works find similar users depending only

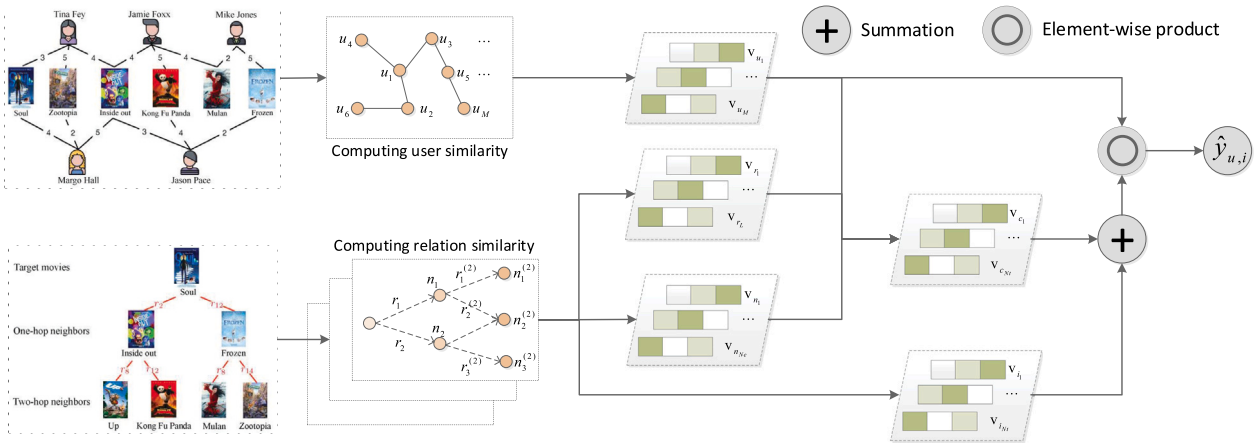


Fig. 3. The framework of the proposed ERSIF-KR.

on interacted items in the user–item interaction graph. Meanwhile, the un-interacted items are typically regarded as negative (disliked) samples (Chen et al., 2019; Wang, He, Cao et al., 2019). However, the unrated items may have never been exposed to the user, rather than the dislike or unintended items. In other words, unobserved interactions do not always represent negative preference. This phenomenon is usually called *exposure bias* (Chen, Dong, Wang et al., 2021), which is very common in recommender systems and might severely deteriorate the recommendation performance (Chen, Dong, Wang et al., 2021; Saito et al., 2020). Hence, we try to mitigate the exposure bias by exploiting indirect interactions and essential differences of items for filtering similar users of the target user. Here we first present how to compute the user similarity degree by alleviating exposure bias and then give the user embedding optimization method.

4.1.1. User similarity computation

Although there are numerous items, only a part of items can be exposed to users. The unobserved feedback might be attributed to unknown or unexposed items rather than dislike ones. In addition to common items rated by both users, the items rated by only one user can also be taken into account for computing the similarity degree of the two users. In other words, for item i_a rated by user u_l and item i_b rated by user u_m, if the two items have a high similarity degree, both users may have a common interest related to two items. Therefore, they should be considered when computing user similarity scores. In addition, the contribution of the two items for user similarity computation should be determined by the correlation of the two items and the difference of their ratings. Based on the above observation, we propose an indirect feedback-based user similarity computation method that considers the indirect interactions and the rating differences of items.

Specifically, suppose that $\mathbf{S} \in \mathbb{R}^{M \times M}$ denotes a sparse adjacency matrix, and its element $s_{l,m}$ is the edge weight representing the interest similarity degree of users u_l and u_m. Note that $s_{l,m} \in \{0, 1\}$ is discrete in traditional recommender systems (Wang, He, Wang et al., 2019; Wang et al., 2018). However, $s_{l,m} \in [0, 1]$ is continuous in our model, which is helpful in discovering the mutual influence between users with similar interests. Let $\mathcal{H}_l$ and $\mathcal{H}_m$ represent the sets of the items rated by user u_l and u_m, respectively, and $\mathcal{H}_l^{single}$ is the set of items rated by user u_l but not user u_m, i.e., $\mathcal{H}_l^{single} = \mathcal{H}_l - \mathcal{H}_m$ — which is also applicable for $\mathcal{H}_m^{single}$. We use the following formula to compute the similarity degree of the two users, which considers the common items interacted by both users and the items interacted by only one user:

$$s_{l,m} = \frac{1-\delta}{|\mathcal{H}_l \cap \mathcal{H}_m|} \sum_{i_c \in \mathcal{H}_l \cap \mathcal{H}_m} \alpha_{i_c} . c_{i_c,i_c} + \frac{\delta}{|\mathcal{H}_l^{single}|} \sum_{i_a \in \mathcal{H}_l^{single}} \max_{i_b \in \mathcal{H}_m^{single}} \{\beta_{i_a,i_b} . c_{i_a,i_b}\}, \tag{9}$$

where the first term reflects the similarity degree based on the common-rated items interacted by both users, and the second term denotes the single-rated items interacted by only one user. Here, α_{i_c} and β_{i_a,i_b} refer to the attention weights for the common-rated items and single-rated items individually, and $\delta \in [0, 1]$ is a hyper-parameter to adjust the weight of the two parts. Moreover, c_{i_a,i_b} means the cosine similarity between item embeddings $\mathbf{v}_{i_a}$ and $\mathbf{v}_{i_b}$:

$$\hat{c}_{i_a,i_b} = \frac{\texttt{Accumulate}(\mathbf{v}_{i_a} \circ \mathbf{v}_{i_b})}{\|\mathbf{v}_{i_a}\|_2 \cdot \|\mathbf{v}_{i_b}\|_2}, \tag{10}$$

where `Accumulate` denotes a function that takes the summation of all elements in the vector, and $\circ$ indicates the element-wise product. Since $\hat{c}_{i_a,i_b} \in [-1, 1]$, we reformulate it as $c_{i_a,i_b} = (\hat{c}_{i_a,i_b} + 1)/2 \in [0, 1]$. For the common-rated items, the cosine similarity c_{i_c,i_c} is always 1 since the two vectors are the same.

The attention score α_{i_c} in Eq. (9) is determined by the difference of user ratings, since the gap indicates the interest difference degree of two users. In general, the closer the ratings of two users for a given item, the more similar their preferences. Similarly, a bigger gap might suggest a low similarity score. Hence, we take the difference of user ratings into account to compute the attention score. To avoid personal rating preference biases, we normalize user ratings by the min–max method. Suppose that t_{u_l} is the normalized rating score from user u_l for item i_c. Then α_{i_c} is calculate as follows:

$$\alpha_{i_c} = 1 - \frac{|t_{u_l} - t_{u_m}|}{\max\{t_{i_c}^*, |t_{u_l} - t_{u_m}|\}}, \tag{11}$$

where $t_{i_c}^*$ is the average rating of item i_c. A smaller difference between the ratings of both users for item i_c suggests that both users have closer interest preferences, and correspondingly the attention score α_{i_c} becomes higher. Similarly, for single-rated items rated only by one user, the difference between item ratings and average ratings are taken into account to compute the attention score as follows:

$$\beta_{i_a,i_b} = 1 - \frac{|(t_{u_l} - t_{i_a}^*) - (t_{u_m} - t_{i_b}^*)|}{\max\{t_{i_a}^*, t_{i_b}^*, |(t_{u_l} - t_{i_a}^*) - (t_{u_m} - t_{i_b}^*)|\}}, \tag{12}$$

where $t_{i_a}^*$ and $t_{i_b}^*$ are the average ratings of item i_a and i_b respectively. The closer the ratings of i_a and i_b, the larger the value of β_{i_a,i_b}.

As a result, the edge weight $s_{l,m}$ between user u_l and u_m can be calculated based on Eqs. (9)–(12). Besides, a higher value of $s_{l,m}$ denotes that the two users have more similar interests. Similarly, a lower value means that the two users have less common interests.

To improve the computational efficiency and filter irrelevant users, we set $s_{l,m} = 0$ if it is less than a threshold value α. In implementation, we set $\alpha = 0.2$ in our experiments. If $s_{l,m} >= \alpha$, we regard user u_m as one

of the similar neighbors for user u_l. Meanwhile, we set $p_{l,m}$ as the graph Laplacian norm similar to GCNs (Kipf & Welling, 2017), computed as follows:

$$p_{l,m} = \frac{s_{l,m}}{\sqrt{|\mathcal{T}_l| \cdot |\mathcal{T}_m|}}, l \text{ and } m \in [1, M], \tag{13}$$

where $\mathcal{T}_l$ and $\mathcal{T}_m$ denote the set of similar neighbor of user u_l and u_m, respectively. $\mathbf{P} = \{p_{l,m}\}$ is the Laplacian matrix of the user similarity, which is used to distinguish the weights of mutual influence among users. It is useful in finding some representative neighbors for each user and learn effective user embedding to improve recommendation performance.

4.1.2. User embedding optimization

We utilize the layer-wise propagation rule of GCNs to optimize users' embeddings $\mathbf{V}_{(u)} = [\mathbf{v}_{u_1}, \ldots, \mathbf{v}_{u_M}]^\top \in \mathbb{R}^{M \times D}$:

$$\mathbf{V}_{(u)}^{(k)} = \sigma\left(\mathbf{P}\mathbf{V}_{(u)}^{(k-1)}\mathbf{W}_{(u)}^{(k-1)}\right), k = 0, 1, \ldots, K, \tag{14}$$

where $\mathbf{V}_{(u)}^{(0)} = \mathbf{V}_{(u)}$ and σ denotes the non-linear activation function; $\mathbf{W}_{(u)}^{(k-1)}$ indicates the learnable weight matrix; and K is the number of network layers. Our model leverages user similarity to build a connection between users, leading to more effective users' embeddings.

4.2. Relation similarity-based item embedding

High-order neighbors are generally utilized for enhancing the target item embedding in KG-based recommendation systems (Guo et al., 2022). However, existing works iteratively incorporate the high-order neighbors into the low-order ones, which fails to consider the connections between different-order relations. We argue that these connections have significant influence in computing the target item embedding. To this end, we propose a relation similarity-based item model to improve the item representation by measuring the high-order neighbors' importance. We begin our description by presenting the method of computing the similarity degree of relations. Subsequently, we illustrate how to incorporate neighbor information with the relation similarity for the target item representation learning.

4.2.1. Relation similarity computation

The similarity degree is computed based on the relations' vectors. Specifically, given a user–item pair (u, i), $\mathcal{N}^{(K)}$ and $\mathcal{R}^{(K)}$ denote the K-hop neighbor and the K-order relation sets of item i, respectively. For the real-world knowledge graph, the size of $\mathcal{N}^{(K)}$ may vary greatly among all entities (items). To keep the efficiency of the model's calculation and training, we set $|\mathcal{N}^{(K)}| = Z$ as a configurable constant. That is, $\mathcal{N}^{(K)}$ may contain duplicates if $|\mathcal{N}^{(K)}| < Z$. Here, we take the two-hop neighbor as an example and describe how to compute the relation similarity degree. We highlight that it is easy to extend this computation to high-order neighbors.

We compute the cosine similarity of first-order relation r and second-order relation $r^{(2)}$ as:

$$h_{r,r^{(2)}} = \frac{\texttt{Accumulate}(\mathbf{v}_r \circ \mathbf{v}_{r^{(2)}})}{\|\mathbf{v}_r\|_2 \cdot \|\mathbf{v}_{r^{(2)}}\|_2}, \tag{15}$$

where $h_{r,r^{(2)}} \in [-1, 1]$. Note that r and $r^{(2)}$ form a complete path in KG: $u \xrightarrow{} i \xrightarrow{r} n \xrightarrow{r^{(2)}} n^{(2)}$. Here, a larger value of $h_{r,r^{(2)}}$ means that $\mathbf{v}_r$ and $\mathbf{v}_{r^{(2)}}$ have more similar distributions and features, and, therefore, the 2nd-order neighbor $n^{(2)}$ should play the same – if not more – important role in learning representation of item i. This similarity degree will be used to estimate the importance of the neighbors for accurate item representation calculation.

Then, we normalize the cosine similarity to learn the weight of each 2nd-order relation $r^{(2)}$:

$$\hat{h}_{r,r^{(2)}} = \frac{\exp(\|\frac{1+h_{r,r^{(2)}}}{2}\|_1)}{\sum_{\hat{r}^{(2)} \in \mathcal{R}^{(2)}} \exp(\|\frac{1+h_{r,\hat{r}^{(2)}}}{2}\|_1)}, \tag{16}$$

where $\hat{h}_{r,r^{(2)}} \in [0, 1]$. Here $\hat{r}^{(2)}$ and $r^{(2)}$ have the same first-order neighbor entity n. Since $h_{r,r^{(2)}}$ ranges from -1 to 1, we normalize it to the range $[0, 1]$ by $(1 + h_{r,r^{(2)}})/2$.

4.2.2. Item embedding optimization

After computing relation similarity, the embeddings of one-hop and two-hop neighbors are calculated with the relation similarity as weights. In particular, following the work (Wang, Zhao et al., 2019), we first calculate the influence score of user u for each first-order relation r:

$$f_{u,r} = \frac{\exp(\|\mathbf{v}_u \circ \mathbf{v}_r\|_1)}{\sum_{\hat{r} \in \mathcal{R}^{(1)}} \exp(\|\mathbf{v}_u \circ \mathbf{v}_{\hat{r}}\|_1)}, \tag{17}$$

where $f_{u,r} \in [0, 1]$ characterizes the importance of the first-order relation r to user u. Additionally, $\hat{r}$ and r have the same target item i. After the optimization of the rotation-based KGE method, we compute the first-order neighbor influence embedding of item i as follows:

$$\mathbf{v}_{c^{(1)}} = \sum_{n \in \mathcal{N}^{(1)}} f_{u,r} \cdot \mathbf{v}_n, \tag{18}$$

where $\mathbf{v}_{c^{(1)}} \in \mathbb{R}^D$.

Through the layer-wise propagation rule based on the knowledge graph, the 2nd-order neighbor influence representation of item i is calculated as:

$$\mathbf{v}_{c^{(2)}} = \sum_{n^{(2)} \in \mathcal{N}^{(2)}} f_{u,r} \cdot \hat{h}_{r,r^{(2)}} \cdot \mathbf{v}_{n^{(2)}}, \tag{19}$$

where $\mathbf{v}_{c^{(2)}} \in \mathbb{R}^D$.

A crucial design of KG-based recommender systems is how to combine the first-order and high-order neighbor influence to learn the comprehensive node embedding. In our model, we implement three types of relation aggregators to model the influence of the first-order and second-order neighbors:

- *Concatenation aggregator* Hamilton et al. (2017) concatenates two vectors of $\mathbf{v}_{c^{(1)}}$ and $\mathbf{v}_{c^{(2)}}$ before applying the nonlinear transformation:

$$\mathbf{v}_c = \sigma\left(\texttt{Concatenation}\left(\mathbf{v}_{c^{(1)}}, \mathbf{v}_{c^{(2)}}\right)\mathbf{W}_{(c)} + \mathbf{b}\right), \tag{20}$$

  where $\mathbf{v}_{c^{(1)}} \in \mathbb{R}^D$ and $\mathbf{v}_{c^{(2)}} \in \mathbb{R}^D$; $\mathbf{W}_{(c)} \in \mathbb{R}^{2D \times D}$ represents the weight matrix; and $\mathbf{b}$ is the corresponding bias. Note that $\texttt{Concatenation}$ indicates a function: $\left(\mathbb{R}^D, \mathbb{R}^D\right) \to \mathbb{R}^{2D}$.
- *Multiplication aggregator* Trouillon et al. (2016) takes the element-wise multiplication of two vectors as:

$$\mathbf{v}_c = \sigma\left(\left(\mathbf{v}_{c^{(1)}} \circ \mathbf{v}_{c^{(2)}}\right)\mathbf{W}_{(m)} + \mathbf{b}\right), \tag{21}$$

  where $\mathbf{W}_{(m)} \in \mathbb{R}^{D \times D}$ is the weight matrix and $\circ$ indicates the element-wise product.
- *Sum aggregator* Wang, Zhao et al. (2019) adds the two vectors followed by a nonlinear transformation:

$$\mathbf{v}_c = \sigma\left(\left((1-\psi) \cdot \mathbf{v}_{c^{(1)}} + \psi \cdot \mathbf{v}_{c^{(2)}}\right)\mathbf{W}_{(s)} + \mathbf{b}\right), \tag{22}$$

  where $\mathbf{W}_{(s)} \in \mathbb{R}^{D \times D}$ denotes a weight matrix, ψ is a balancing parameter, and σ denotes the non-linear activation function (i.e., ReLU is used here).

4.3. Model prediction and optimization

To predict the item clicking probability, user representation $\mathbf{v}_u$ and item representation $\mathbf{v}_i$ as well as corresponding neighbor influence representation $\mathbf{v}_c$ are fed into the multiply function:

$$\hat{y}_{u,i} = \texttt{Accumulate}(\mathbf{v}_u \circ (\mathbf{v}_i + \mathbf{v}_c)). \tag{23}$$

We leverage the negative sampling strategy during training to make the effective computation. Correspondingly, the first loss function of our model ERSIF-KR is:

$$\mathcal{L}_{\mathrm{CF}} = \sum_{u \in \mathcal{U}} \left(\sum_{i \in P} C\left(y_{u,i}, \hat{y}_{u,i}\right) - \sum_{\bar{i} \in \overline{P}} C\left(y_{u,\bar{i}}, \hat{y}_{u,\bar{i}}\right) \right), \tag{24}$$

where C indicates the cross-entropy loss function, $P = \{(u,i)|y_{u,i} = 1, u \in \mathcal{U}, i \in \mathcal{E}\}$ and $\overline{P} = \{(u,\bar{i})|y_{u,\bar{i}} = 0, u \in \mathcal{U}, \bar{i} \in \mathcal{E}\}$ are positive and negative sampling distributions, respectively. Note that item i may be the head entity h or the tail entity t in the KG.

For knowledge graph embedding methods, controlling the distance function range can directly affect the final negative sampling loss (Zhang et al., 2020). Meanwhile, a suitable distance function can help alleviate the over-fitting problem (Sun et al., 2019). Therefore, we leverage the Radial Basis Function (RBF) to optimize the distance function of entities and relations:

$$d^{(1)} = \exp(-\gamma_1 \|\mathbf{W}_{(b_1)} \cdot (\mathbf{v}_i \circ \mathbf{v}_r - \mathbf{v}_n)\|^2), \tag{25}$$

$$d^{(2)} = \exp(-\gamma_2 \|\mathbf{W}_{(b_2)} \cdot (\mathbf{v}_n \circ \mathbf{v}_{r^{(2)}} - \mathbf{v}_{n^{(2)}})\|^2), \tag{26}$$

where $\gamma_1 > 0$ and $\gamma_2 > 0$ are two hyper-parameters, $\mathbf{W}_{(b_1)} \in \mathbb{R}^{D \times D}$ and $\mathbf{W}_{(b_2)} \in \mathbb{R}^{D \times D}$ denote two weight matrices. The range of a single RBF kernel is between $[0, 1]$, $d^{(1)} \in (0, 1]$ and $d^{(2)} \in (0, 1]$. The larger value of $d^{(1)}$ (or $d^{(2)}$) indicates a shorter distance between the two entities. The complete distance function of rotation-based KGE method becomes:

$$d(h, r, t) = \frac{1}{d^{(1)} + d^{(2)} + \beta}, \tag{27}$$

where β is a constant used to avoid dividing by 0.

Similar to prior works (Bordes et al., 2013; Wang, He, Cao et al., 2019) that consider the relative order between true triplets and broken ones, we use the pairwise ranking loss to encourage their discrimination. For each true triplet (h, r, t) and broken triplet (h, r, t'), the second loss function of our model ERSIF-KR is:

$$\mathcal{L}_{\mathrm{KG}} = \sum_{(h,r,t,t') \in \mathcal{K}} -\log \mathrm{Sigmoid}\left(d(h, r, t') - d(h, r, t)\right), \tag{28}$$

where a lower value of $d(h, r, t)$ (or $d(h, r, t')$) suggests that the triplet (h, r, t) (or (h, r, t')) is more likely to be true. $\mathcal{K} = \{(h, r, t, t')|(h, r, t) \in \mathcal{G}, (h, r, t') \notin \mathcal{G}\}$. (h, r, t') is a reconstructed triplet by replacing one entity in a set of entities that do not have triple connectivity based on r and h.

Lastly, the total loss function of our model combines Eqs. (24) and (28) with a regularizer term:

$$\mathcal{L}_{\text{ERSIF-KR}} = \mathcal{L}_{\mathrm{CF}} + \mathcal{L}_{\mathrm{KG}} + \lambda \|\Theta\|_2^2, \tag{29}$$

where the last term is the L2 regularization, and Θ denotes the model's parameter. λ represents the L2 regularizer weight, which is used to alleviate the overfitting problem. As shown in Eq. (29), the loss function of our model is mainly composed of the prediction loss L_{CF} and the knowledge graph embedding loss L_{KG}, whose training processes are summarized in Algorithm 1 and 2, respectively. During the process of training ERSIF-KR, L_{CF} and L_{KG} are optimized alternatively, where mini-batch Adam (Kingma & Ba, 2015) algorithm is adopted to train the model.

4.4. Discussions

In this section, we first show the connection between our model and previous KG-based recommender system such as KGAT (Wang, He, Cao et al., 2019). Next, we analyze the time complexity of our model ERSIF-KR.

Algorithm 1 The process of optimizing loss $\mathcal{L}_{CF}$.

Input: User set $\mathcal{U}$, item set $\mathcal{E}$, user-item interaction matrix $\mathbf{Y}$, and user-item rating matrix $\mathbf{T}$.
Ensure: Predicting matrix $\hat{\mathrm{Y}}$
1: Initializing user matrices $\mathbf{V}_{(u)}$, item matrices $\mathbf{V}_{(i)}$, and the learnable weight matrix $\mathbf{W}_{(u)}^{(*)}$;
2: **for** num_epoch = 0, ... , N_{epoch} **do**
3: Compute item similarity $\hat{c}_{i_a,i_b}$, attention score α_{i_c} and β_{i_a,i_b} via Eqs. (10), (11), and (12);
4: Balance the relative weight between above three parts to calculate the user similarity $s_{l,m}$ via Eq. (9);
5: Calculate the graph Laplacian norm value $p_{l,m}$ via Eq. (13);
6: Updating the learnable weight matrix $\mathbf{W}_{(u)}^{(*)}$ and optimize the user representation matrix $\mathbf{V}_{(u)}^{(k)}$ via Eq. (14);
7: Predicting the clicking probability of user u for item i via Eq. (23).
8: **end for**

Algorithm 2 The optimization of knowledge graph embedding loss $\mathcal{L}_{KG}$.

Input: User set $\mathcal{U}$, item set $\mathcal{E}$, and knowledge graph triple set (h, r, t).
Ensure: Minimizing the distance $d(h, r, t)$ of each knowledge graph triple set (h, r, t)
1: Optimize item representation matrices $\mathbf{V}_{(i)}$, entity representation matrices $\mathbf{V}_{(n)}$, and relation representation matrices $\mathbf{V}_{(r)}$ via Eqs. (1) and (2);
2: **for** num_epoch = 0, ... , N_{epoch} **do**
3: Calculate the cosine similarity $h_{r,r^{(2)}}$ and user preference score $f_{u,r}$ via Eqs. (15) and (17);
4: Calculate the first-order and second-order neighbor influence representations via Eqs. (18) and (19);
5: Combine the multi-hop neighbor influence representations via one of Eqs. (20)–(22).
6: Compute the distance $d(h, r, t)$ of each knowledge triplet set (h, r, t) via Eqs. (25)–(27).
7: **end for**

4.4.1. Relation to previous KG-based recommender systems

To consider multi-hop neighbors for enhancing item embedding, existing KG-based recommendation models such as RippleNet (Wang et al., 2018), KGCN (Wang, Zhao et al., 2019), KGAT (Wang, He, Cao et al., 2019) and AKUPM (Tang et al., 2019) explicitly model the high-order connectivities in KG. These models recursively propagate the embeddings from neighbors to refine the node embedding, e.g., the propagation in KGAT is defined as:

$$v_h^{(l)} = f(v_h^{(l-1)}, v_{\mathcal{N}_h}^{(l-1)}) \tag{30}$$

wherein the information propagated within l-ego network for the entity h is defined as:

$$v_{\mathcal{N}_h}^{(l-1)} = \sum_{(h,r,t) \in \mathcal{N}_h} \pi(h, r, t) v_t^{(l-1)} \tag{31}$$

where $v_t^{(l-1)}$ is the representation of entity t generated from the $(l-2)$-th propagation step, memorizing the information from its $(l-1)$-hop neighbors; $\mathcal{N}_h$ is the corresponding knowledge graph triplet set of node h; $v_h^{(0)}$ is set as v_h at the initial information-propagation iteration. When considering two-hop neighborhood, this method incorporates the embedding of two-hop neighbors into the one-hop one, which is further incorporated into the target item embedding.

In our model ERSIF-KR, the embeddings of two-hop neighbors are directly incorporated into the target item embedding with a weight. Aligning with Eqs. (2) and (19), we can see that ERSIF-KR is equivalent to KGAT by setting the weights and aggregation function accordingly. The main difference is the weight factor used to determine the

Table 2
Statistics and hyper-parameter settings of three datasets.

	Last.FM	MovieLens-20M	Dianping-Food
Users	1872	138,159	2,298,698
Items	3846	16,954	1,362
Interactions	42,346	13,501,622	23,416,418
Density	0.59%	0.58%	0.75%
Entities	9,366	102,569	28,115
Relations	60	32	7
KG triplets	15,518	499,474	160,519

importance of multi-hop neighborhood in enhancing the target item embedding, which can be adjusted to the same by changing corresponding parameters. Another difference from previous KG-based models is that our ERSIF-KR takes into account exposure bias to enhance the process of finding similar users, which is further used to refine the target entity embedding.

4.4.2. Time complexity analysis

The computational cost of ERSIF-KR mainly comes from following three parts. The first one is the indirect feedback-based user module, requiring $O(N_t \cdot N_t)$ time complexity (cf. Eq. (10)), where N_t denotes the number of items. For the knowledge graph embedding part, the translation principle has computational complexity $O(max\{N_i, N_e\} \cdot L)$ in Eq. (1) and (2). In the final prediction layer, only the inner product is involved and its time complexity is $O(M \cdot N_t)$ as presented in Eq. (23). Therefore, the total complexity for training our ERSIF-KR is $O(N_t \cdot N_t + max\{N_i, N_e\} \cdot L + M \cdot N_t)$.

5. Experimental evaluations

In this section, we evaluate our proposed ERSIF-KR model on three real-world scenarios including movie, music, and restaurant recommendations. Furthermore, we report and discuss our observations from extensive performance evaluation and comparison with the state-of-the-art baselines, followed by model ablation studies and parameter sensitivity investigation. We also present a case study to explain the significance of our proposed method.

5.1. Experimental settings

5.1.1. Datasets

To compare our method's performance against several baselines, we conduct comprehensive experiments on the three commonly used benchmark datasets, i.e., Last.FM, MovieLens-20M, and Dianping-Food. The statistics of the three datasets are summarized in Table 2.

- **Last.FM**[2] consists of music listening records of approximately 2000 users from a popular music website Last.Fm.
- **MovieLens – 20M**[3] is a widely used dataset for movie recommendations, which has nearly 20 million ratings (ranging from 1 to 5) w.r.t. 27,278 movies on the MovieLens website.
- **Dianping – Food**[4] is a dataset collected by Wang, Zhang, Zhang et al. (2019), consisting of over 10 million interactions, including clicking, purchasing, and adding to favorites, between nearly 2 million users and 1000 restaurants in Dianping.com from May 1, 2015 to December 12, 2018.

Following prior related studies (Wang, Zhang, Zhang et al., 2019; Wang, Zhao et al., 2019), Microsoft Satori[5] is used to construct the knowledge graphs for Last.FM and MovieLens-20M datasets. In addition, all triples' subsets are selected from the whole KG with a confidence level greater than 0.9. For each sub-KG, Satori IDs are collected from all valid musics/movies by matching their names with the tail of triples, i.e., (head, music.artist.origin, tail), (head, film.film.actor, tail), or (head, type.object.name, tail). For simplicity, items with multiple matched or no matched entities are deleted. The item IDs are matched with the head of all triples, and well-matched triples are selected as the final KG for Last.FM and MovieLens-20M. The KG of Dianping-Food is collected from Dianping.com, which is a Chinese group buying website including consumer reviews of restaurants, and built by the internal toolkit (Meituan Brain) of Meituan-Dianping Group. This dataset contains some richer types, e.g., restaurant, city, first-level and second-level category, star, business area, and tag of entities.

5.1.2. Evaluation metrics

As one metric might not adequately reflect the usefulness for practical applications, we adopt two widely used metrics to evaluate the performance of our proposed ERSIF-KR model as well as the baseline models and leave other evaluation metrics for future exploration. (1) For click-through rate (CTR) prediction, the trained model is utilized to predict each user–item interaction in the test set. We use *AUC* and *F-score (F1)* metrics to evaluate the CTR prediction. (2) For top-K recommendation, the trained model selects top-K items with the highest predicted click probability for each user in the test set. We choose the *Recall@K (R@K)* metric to evaluate top-K recommendation.

5.1.3. Baselines

To demonstrate the effectiveness and superiority of the proposed model, we compare our ERSIF-KR with graph-based methods (e.g., GC-MC Berg et al., 2018 and NGCF Wang, He, Wang et al., 2019), path-based methods (e.g., MCRec Hu et al., 2018 and RippleNet Wang et al., 2018), and KG-based methods (e.g., KGAT Wang, He, Cao et al., 2019, KGNN-LS Wang, Zhang, Zhang et al., 2019, KGIN Wang et al., 2021c, and CKAN Wang et al., 2020a).

- **GC-MC** (Berg et al., 2018): is a graph-based framework, which applies the GCNs (Kipf & Welling, 2017) on the user–item interaction graph. Here we employ it on the user–item knowledge graph, and use one graph convolution layer suggested in (Berg et al., 2018).
- **NGCF** (Wang, He, Wang et al., 2019): is a CF-based and graph-based framework, which exploits the user–item graph structure by propagating the embeddings of users and items.
- **RippleNet** (Wang et al., 2018): is a classical embedding-based and path-based method that propagates user preferences on the knowledge graph and enriches user representation with relevant items for each user.
- **CIEPA** (Lin et al., 2021): is a novel knowledge-enhanced recommendation framework, which incorporates the knowledge graph into a co-liked co-occurrence matrix with path-level attention.
- **KGAT** (Wang, He, Cao et al., 2019): is a hybrid KG-based model with the attention mechanism and TransR method, optimizing the node's embedding by recursively propagating its embedding from its neighbors' embeddings.
- **KGNN-LS** (Wang, Zhang, Zhang et al., 2019): is a novel KG-based model with label smoothness regularization, which computes personalized user-specific item embeddings by identifying important relations for a specific user in KG. Moreover, it also provides regularization over the edge weights and better inductive bias through label smoothness.

[2] https://grouplens.org/datasets/hetrec-2011/
[3] https://grouplens.org/datasets/movielens/
[4] https://github.com/hwwang55/KGNN-LS
[5] https://searchengineland.com/library/bing/bing-satori

Table 3
Hyper-parameter settings of our model on three datasets.

	Last.FM	MovieLens-20M	Dianping-Food
Batch size	2048	10,240	10,240
α	0.2	0.2	0.2
K	1	2	3
Z	2	8	8
D	32	64	64
δ	0.7	0.7	0.7
η	9×10^{-4}	8×10^{-4}	1.5×10^{-3}
λ	6×10^{-5}	6×10^{-8}	1×10^{-4}
γ_1	50	70	100
γ_2	70	110	130

- **KGIN** (Wang et al., 2021c): is a recently proposed KG-enhanced recommendation model, which identifies latent intention of users, and further performs the relational path-aware aggregation for both user-intent-item and KG triplets.
- **CKAN** (Wang et al., 2020a): is the collaborative knowledge-aware attentive network, which adopts a heterogeneous propagation approach to encode both kinds of information and uses a knowledge-aware attention mechanism to distinguish contributions of different knowledge-based neighbors.

5.1.4. Parameter settings

We implement our model ERSIF-KR with Tensorflow on a machine with an NVIDIA GeForce GTX 3090. For each dataset, similar to most KG-based recommendation methods (Sun et al., 2020), we randomly split it into the training set (60%), validation set (20%), and test set (20%). Specifically, we optimize the training process by leveraging Adam optimizer (Kingma & Ba, 2015). The hyper-parameters of ERSIF-KR for the three datasets (music, movie, and restaurant) are determined by optimizing R@10 on the validation set, and the detailed settings are shown in Table 3. Specifically, K represents the dimensions of hidden layers in user embedding; Z denotes the neighbor sampling size; D is the dimensions of embedding for users, relations, and items; η indicates the learning rate; λ represents the L2 regularizer weight; ψ is the relative weight parameter between the first-order and second-order neighbor influence representations; and $\beta = 0.0001$ is a constant parameter. For the baselines, we also adopt the training set and validation set to tune their hyper-parameters for the following experiments.

5.2. Performance comparison

In Table 4, we report the results of the comparisons between ERSIF-KR and baseline models across three datasets in the context of top-K recommendation and CTR prediction. From these results, we have following observations.

First, ERSIF-KR consistently yields better performance in terms of the two evaluation metrics on all the datasets. For example, compared to KGIN, ERSIF-KR-sum exhibits improvements of 2.8%, 2.4%, and 3.2% on the Last.FM, MovieLens-20M, and Dianping-Food datasets, respectively, in terms of *Recall@10* metric. The substantial improvement of our model over baselines verifies the significance of capturing the indirect interaction when finding similar users and relation similarity for measuring the weight of the neighbors for a target item. Besides, ERSIF-KR-sum outperforms the other two aggregation methods (ERSIF-KR-con and ERSIF-KR-mul), which indicates that the simple addition aggregator is the best effective approach in merging neighboring entities.

Second, KG-based methods (e.g., KGAT, KGNN-LS, KGIN and CKAN) achieve better performance than graph-based and path-based approaches across all datasets. The reason behind is that the latter only considers item embedding optimization with respect to structural, textual, and visual knowledge, but fails to fully consider the importance of user preference and the connectivity between relations and items. Instead, KGAT models the high-order connectivities and refines a target node's embedding by propagating the embeddings in the target node's neighbors within a KG. Meanwhile, KGAT utilizes the attention mechanism to learn the importance of the neighbors, and thus performs better. Furthermore, KGNN-LS models the high-order connectivity between entities based on the relations and identifies important relations in the KG for a given user. It leverages the label smoothness method to regularize the edge weights on a user-specific weighted graph. Hence, KGNN-LS achieves better performance compared to KGAT.

Recently, KGIN considers identifying latent intention of users, and further performs the relational path-aware aggregation for both user-intent-item and KG triplets. CKAN combines explicitly encoded collaborative signal latent in user–item interactions with auxiliary knowledge associations in KG together. Hence KGIN and CKAN obtain higher accuracy than KGNN-LS. However, ERSIF-KR achieves the best performance than the above-mentioned KG-based methods, which proves our motivation of efficiently exploiting the indirect feedback of users for alleviating exposure bias as well as the similarity degree of different-order relations for identifying the importance of multi-hop neighbors. Also, we conduct t-tests to investigate whether the improvements are statistically significant. The results are presented in Table 4, where the improvements that are statistically significant are marked with an asterisk ("*"). As shown, our model ERSIF-KR-sum is significantly better than the best baseline model in general.

5.3. Ablation study

Now we focus on studying the contributions of two essential components in our ERSIF-KR model, i.e., the indirect feedback-based user module and the relation similarity-based item module. We evaluate the two components' effectiveness by measuring the performance of the variant when one of the component is disabled. Specifically, we study the role of different component by considering following variants of ERSIF-KR:

- **ERSIF-KR-Base** predicts the probability as: $\hat{y}_{u,i} = \texttt{Accumulate}\left(\mathbf{v}_u \circ \mathbf{v}_i\right)$ without involving the two components mentioned above.
- **ERSIF-KR-User** only consists of the indirect feedback-based user module and predicts the probability as: $\hat{y}_{u,i} = \texttt{Accumulate}\left(\mathbf{v}_u \circ \mathbf{v}_i\right)$. We respectfully note that $\mathbf{v}_u$ in ERSIF-KR-user is different from $\mathbf{v}_u$ in ERSIF-KR-base.
- **ERSIF-KR-NoRe** leverages both first-order and second-order neighbor influence, but without considering the connections between different-order relations for item embedding.
- **ERSIF-KR-Re** considers both first-order and second-order neighbor influence and involves the relation similarity-based item module to enhance the target item embedding.
- **ERSIF-KR** is the full version incorporating all the components such as the indirect feedback-based user module and relation similarity-based item module.

We summarize the results reported in Table 5 as follows:

(i) ERSIF-KR performs the best, while ERSIF-KR-Base is the worst model, which implies the effectiveness of the main components we proposed can significantly improve the recommendation performance.

(ii) ERSIF-KR-User achieves better performance than ERSIF-KR-base. The reason is that user similarity degree computed by considering indirect feedback can alleviate exposure bias and accurately reflect the mutual influence between a user and others who have similar preferences. Additionally, modeling the user similarity is beneficial in optimizing user embedding, which may also improve downstream recommendation performance.

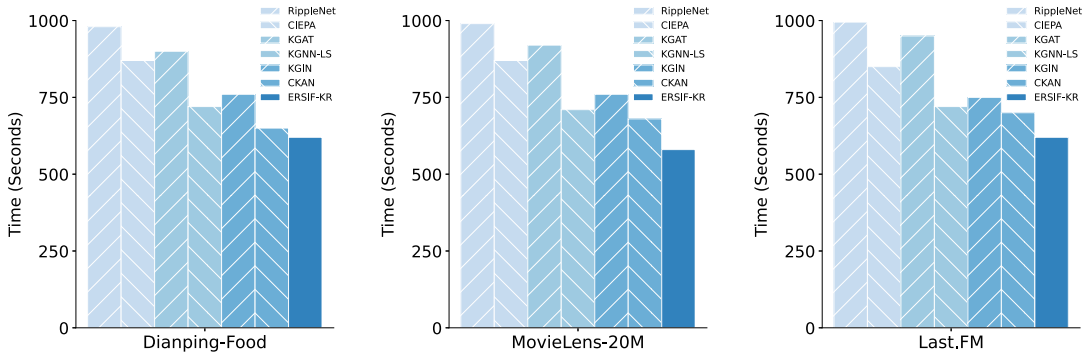
(iii) By incorporating the influence of neighboring entities, both ERSIF-KR-NoRe and ERSIF-KR-Re outperform ERSIF-KR-Base and ERSIF-KR-User. This result confirms the advantages of considering the influence of neighboring entities which is the key factor in KG-based methods. Besides, ERSIF-KR-Re outperforms ERSIF-KR-NoRe, implying that taking the connections between different-order relations into account is essential in enhancing recommendation results.

Table 4
Performance comparison on three datasets.

Method	Last.FM					MovieLens-20M					Dianping-Food				
	R@10	R@50	R@100	AUC	F1	R@10	R@50	R@100	AUC	F1	R@10	R@50	R@100	AUC	F1
GC-MC	0.108	0.253	0.341	0.781	0.681	0.121	0.262	0.420	0.967	0.917	0.149	0.317	0.436	0.841	0.797
NGCF	0.111	0.260	0.353	0.785	0.684	0.139	0.297	0.439	0.971	0.921	0.156	0.325	0.451	0.843	0.803
RippleNet	0.101	0.242	0.336	0.770	0.679	0.130	0.278	0.447	0.973	0.924	0.155	0.328	0.440	0.845	0.814
CIEPA	0.105	0.246	0.339	0.774	0.684	0.134	0.285	0.451	0.977	0.929	0.158	0.332	0.445	0.849	0.818
KGAT	0.117	0.269	0.361	0.792	0.691	0.147	0.313	0.446	0.976	0.925	0.162	0.331	0.472	0.847	0.821
KGNN-LS	0.122	0.277	0.370	0.803	0.692	0.155	0.321	0.458	0.979	0.929	0.170	0.340	0.487	0.850	0.826
KGIN	0.138	0.348	0.434	0.851	0.773	0.169	0.350	0.486	0.984	0.938	0.182	0.360	0.518	0.880	0.843
CKAN	0.133	0.339	0.429	0.842	0.769	0.161	0.342	0.476	0.976	0.929	0.176	0.357	0.506	0.878	0.832
ERSIF-KR-con	0.136	0.347	0.434	0.849	0.773	0.163	0.345	0.481	0.983	0.937	0.179	0.362	0.508	0.878	0.847
ERSIF-KR-mul	0.134	0.344	0.427	0.847	0.770	0.168	0.349	0.487	0.981	0.933	0.182	0.368	0.521	0.876	0.843
ERSIF-KR-sum	0.142*	0.351*	0.442*	0.853	0.775	0.173*	0.355*	0.496*	0.985	0.942*	0.188*	0.371*	0.529*	0.882	0.850*

Table 5
The results of ablation study.

Method	Last.FM			MovieLens-20M			Dianping-Food		
	R@10	AUC	F1	R@10	AUC	F1	R@10	AUC	F1
ERSIF-KR-Base	0.045	0.651	0.562	0.063	0.912	0.875	0.089	0.768	0.743
ERSIF-KR-User	0.103	0.764	0.685	0.116	0.943	0.906	0.146	0.826	0.812
ERSIF-KR-NoRe	0.126	0.812	0.731	0.151	0.965	0.927	0.170	0.854	0.835
ERSIF-KR-Re	0.131	0.837	0.753	0.162	0.974	0.933	0.175	0.869	0.841
ERSIF-KR	**0.142**	**0.853**	**0.775**	**0.173**	**0.985**	**0.942**	**0.188**	**0.882**	**0.850**



(a) Dianping-Food. (b) MovieLens-20M. (c) Last.FM.

Fig. 4. Model efficiency comparisons on three datasets.

5.4. Model efficiency

Now we turn to investigate the model efficiency of the representative KG-based methods on the convergence of model training. As shown in Fig. 4, our model ERSIF-KR requires significantly less running time compared to the traditional (i.e., CKE and PER) and the recent proposed models (RippleNet, KGAT, KGNN-LS, HAGERec, and CKAN). Taking MovieLens-20M for example, the consumed time of our model is around 15.1% lower than the most efficient baseline CKAN, which demonstrates that ERSIF-KR exhibits higher efficiency when the size of the KG is large (e.g., nearly half a million triplets in MovieLens-20M). The possible reasons are as follows. (1) For the indirect feedback-based user module, we compute the user similarity based on the user–item interaction graph independent of the model's training process. (2) For the relation similarity-based item module, the rotation-based KGE method has been shown to be more scalable and effective in prior studies (Sun et al., 2019; Zhang et al., 2020). Following CS-GNN (Hou et al., 2020), we only leverage some simple and efficient aggregators (i.e., sum and concatenation) to model the influence of entity neighbors. Hence, our model achieves higher computational and training efficiency.

5.5. On cold start recommendation

One main reason for incorporating KG into recommender systems is to alleviate the sparsity and cold start problems. We build the data sparsity scenarios by varying the ratio of the training set on the three datasets while keeping the validation set and test set fixed. Table 6 presents the AUC results for three sparsity situations, i.e., 10%, 55%, and 100% of the training set. From experimental results, we can observe that for the eight baselines, the performance using 10% of the training data significantly decreases, compared with the case using full training data. For example, the AUC results of CKAN – the best baseline approach – averagely drops 5.24% using 10% of the training data on the three datasets. However, the performance of ERSIF-KR only decrease 3.18%, which proves that our model can achieve robust performance when user–item historical interaction data is sparse. One important reason is that ERSIF-KR can efficiently exploit indirect feedback to alleviate the exposure bias and fully incorporate entity neighbor influence from KG into target item representation learning.

5.6. Hyper-parameter sensitivity

Next, we investigate the influence of several important parameters in our model, including the weight parameter δ for computing user

Table 6
AUC results of addressing the cold start problem.

Method	Last.FM			MovieLens-20M			Dianping-Food		
	10%	55%	100%	10%	55%	100%	10%	55%	100%
GC-MC	0.740	0.755	0.773	0.916	0.939	0.967	0.812	0.823	0.841
NGCF	0.746	0.765	0.787	0.937	0.956	0.972	0.816	0.825	0.842
RippleNet	0.739	0.756	0.770	0.935	0.959	0.973	0.812	0.825	0.845
CIEPA	0.744	0.760	0.774	0.939	0.963	0.977	0.817	0.828	0.849
KGAT	0.768	0.783	0.796	0.941	0.958	0.976	0.819	0.828	0.845
KGNN-LS	0.776	0.791	0.803	0.952	0.961	0.979	0.824	0.832	0.850
KGIN	0.808	0.830	0.851	0.963	0.971	0.984	0.836	0.859	0.880
CKAN	0.780	0.823	0.842	0.955	0.967	0.976	0.827	0.841	0.878
ERSIF-KR	**0.818**	**0.833**	**0.853**	**0.968**	**0.975**	**0.985**	**0.852**	**0.868**	**0.882**

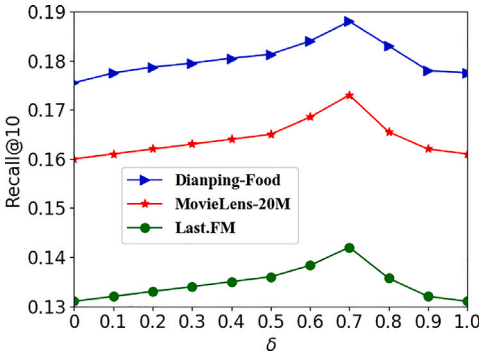


Fig. 5. Influence of δ — the weight parameter in computing user similarity.

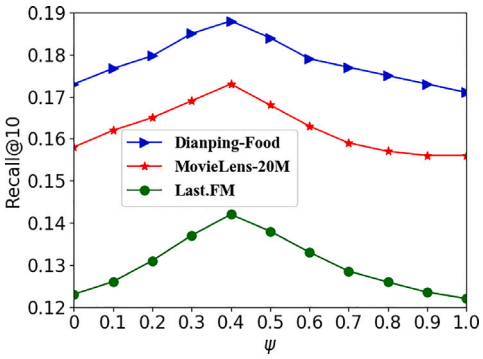


Fig. 6. The effect of the relative weight ψ.

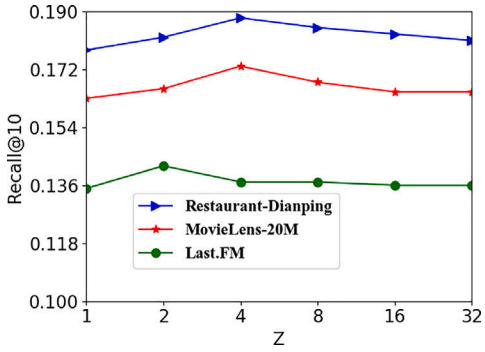


Fig. 7. The effect of the neighbor sampling size Z.

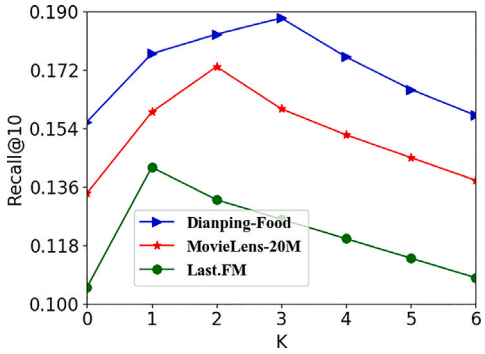


Fig. 8. The effect of the number of hidden layer K.

similarity, the relative weight of different-order neighbors, the sampled neighbor size and the number of hidden layers.

5.6.1. Impact of δ in computing user similarity

We first evaluate the impact of weight parameter δ for computing user similarity in Eq. (9). δ is used to adjust the weight of the two parts, i.e., the common items rated by both users and the items only rated by a single user. As shown in Fig. 5, ERSIF-KR achieves the better performance when $\delta = 0.7$ on the three datasets, which denotes that the latter part has a more important impact than the former one in computing user similarity, because the latter can effectively reflect the user similarity degree by considering the items that are not directly interacted by both users to alleviate the exposure bias. Generally, the number of items rated by a single user is higher than those rated by both users. Exposure bias is a representative bias in existing recommender systems, which brings the ambiguity in the interpretation of unobserved interactions. Meanwhile, the inability to

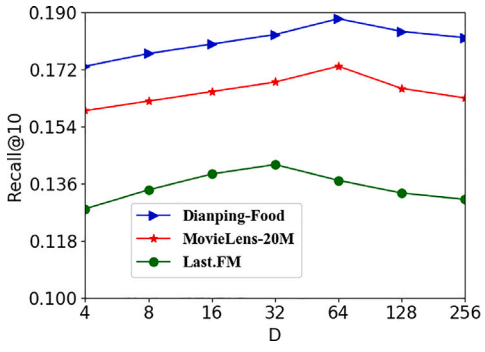


Fig. 9. The effect of the dimensions of embedding D.

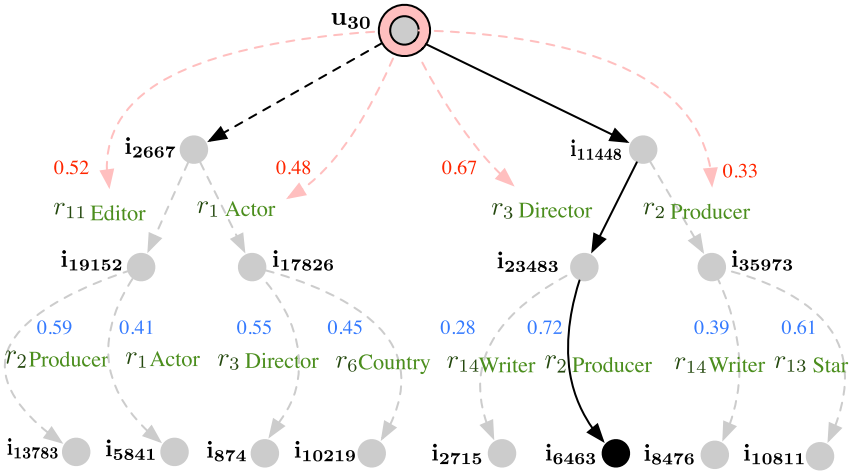


Fig. 10. A case study on MovieLens-20M.

distinguish real negative interactions (e.g., exposed but uninterested) from the potentially positive ones (e.g., unexposed) will result in severe biases.

5.6.2. Impact of ψ — the relative weight of different-order neighbors

Then, we discuss the effect of hyper-parameter ψ in Eq. (22). This parameter denotes the relative weight between the first-order and second-order neighbor influence vectors, which plays a crucial role in balancing the influence of neighbors in target item embedding. Fig. 6 shows that the influence of ψ on Recall@10 results. The performance is inferior when ψ is small (e.g., $\psi = 0$) or large (e.g., $\psi = 1$). ERSIF-KR achieves the best performance when ψ is close to an intermediate value, i.e., $\psi = 0.4$. This suggests that both first-order and second-order neighbor influence representations contribute to the improvement of recommendation performance.

5.6.3. Impact of Z — the sampled neighbor size

Subsequently, we vary the sampled neighbor sizes Z to study the efficacy of KG. According to Fig. 7, ERSIF-KR achieves the best performance when $Z = 2$ or 4 in different datasets. The main reason is that there are various average number of edges in these datasets. Specifically, the average number of edges of each node on the Last.FM, Movielens-20M, Dianping-Food datasets are 1.66, 4.87, and 5.71, respectively. A larger value of Z would increase the number of duplicate neighbors, which do not contribute meaningful neighbors to our model's performance on representation learning as well as the downstream recommendations.

5.6.4. Impact of K — the number of hidden layers

The results of changing the number of hidden layers K are illustrated in Fig. 8. We can see that K is very sensitive to the recommendation performance, and ERSIF-KR achieves the best performance at different K values in different datasets. One important reason is that the three datasets have different numbers of users, and, we can usually obtain better user representation as the number of users increases. Meanwhile, further increasing K would deteriorate the model performance, demonstrating that a neural network with appropriate depth is more conducive to learning entity embeddings.

5.6.5. Impact of D — the embedding dimensions

Finally, we investigate the influence of embedding dimensions D on the performance of ERSIF-KR. The results are shown in Fig. 9, which indicates that a larger value of D can capture richer information of users, entities, and relations. However, an extremely larger value of D such as 128 and 256 would inevitably increase the model complexity and subsequently lead to overfitting problem.

5.7. Case study

When modeling the relation of similarity-based items, the neighbors are aggregated by employing users' influence scores and relation similarity degrees to enhance item representation. To offer reasonable explanations w.r.t. the roles of the two factors, we randomly selected one user (u_{30} here) from the MovieLens-20M and the relevant item i_{11448} from the testing set. Fig. 10 shows the visualization of the relation similarity-based modeling, which explains why we recommend i_{11448} to user u_{30} when user u_{30} has watched movie i_{6463} in the training set.

In Fig. 10, the four red numbers mean the user influence scores for the first-order relation. Among them, the value 0.67 is the largest one. According to the data analysis, user u_{30} has a preference for the movies directed by the director of movie i_{11448} and i_{23483}. Here, the director is the same for both movies. The eight blue numbers refer to the relation similarity degrees between the first-order and second-order relations. Among them, the value 0.72 is the biggest one, since the common producer of movies i_{23483} and i_{6463} has a long-term and stable cooperation with the director of movie i_{11448}, and they have a close correlation value. Meanwhile, the path $u_{30} \rightarrow i_{11448} \xrightarrow{\text{Director}} i_{23483} \xrightarrow{\text{Producer}} i_{6463}$ has the highest weight score, which is represented by solid lines in the figure. Hence, in terms of the product of the two weight values, movie i_{11448} is recommended to user u_{30}. The final clicking probability is predicted based on multiple factors: the user influence scores, the relation similarity degree, the neighbor, as well as the embeddings of users and target items. Here, we only consider the first two weights and state their contributions in the recommendation system, and the final recommended result is determined by Eq. (23).

6. Conclusion

In this paper, we presented a novel knowledge graph enhanced recommender system, called ERSIF-KR, which exploits the indirect feedback and the diversity of the multi-hop neighbors for recommendation. In particular, the specifically designed user module can explore indirect feedback of items that are not directly interacted with users to alleviate the exposure bias while enhancing user representations. In terms of item embedding, we proposed a relation similarity-based item module that directly incorporates multi-hop neighbors into the target item embedding with weights to enrich item representations. These modules can be applied to most existing deep recommendation models to improve the efficiency of user and item representation learning. The experimental results on real-world datasets demonstrate that ERSIF-KR outperforms the state-of-the-art baselines in terms of recommendation performance and model efficiency, while effectively alleviating the sparsity and cold start problems in existing studies.

In the future, we plan to incorporate rich auxiliary data, such as user review and item description, to further enhance the proposed recommendation model. Moreover, we are interested in applying the designed model to other recommendation scenarios, including but not limited to music recommendation, news recommendation, session-based/sequential recommendation.

CRediT authorship contribution statement

Zhonghai He: Resources, Methodology, Data curaton, Writing – review & editing. **Bei Hui:** Methodology, Resources, Writing – review & editing . **Shengming Zhang:** Sofware, Validation, Investigation. **Chunjing Xiao:** Conceptualizaton, Methodology, Writing – original draft. **Ting Zhong:** Conceptualizaton, Methodology, Resources. **Fan Zhou:** Writing – review & editing, Resources, Formal analysis.

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.

Data availability

Data will be made available on request.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant No. 62176043 and 62072077), Natural Science Foundation of Sichuan Province (Grant No. 2022NSFSC0505), and Sichuan Science and Technology Program (Grant No. 2022YFSY0006).

References

Abu-El-Haija, S., Perozzi, B., Kapoor, A., Alipourfard, N., Lerman, K., Harutyunyan, H., Ver Steeg, G., & Galstyan, A. (2019). Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *International conference on machine learning* (pp. 21–29).

Ai, Q., Azizi, V., Chen, X., & Zhang, Y. (2018). Learning heterogeneous knowledge base embeddings for explainable recommendation. *Algorithms*, *11*(9), 137.

Berg, R. v. d., Kipf, T. N., & Welling, M. (2018). Graph convolutional matrix completion. In *Proceedings of the ACM SIGKDD International conference on knowledge discovery and data mining* (pp. 1–7).

Bordes, A., Usunier, N., García-Durán, A., Weston, J., & Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems* (pp. 2787–2795).

Cao, Y., Wang, X., He, X., Hu, Z., & Chua, T. (2019). Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *The world wide web conference* (pp. 151–161).

Chen, J., Dong, H., Qiu, Y., He, X., Xin, X., Chen, L., Lin, G., & Yang, K. (2021). Autodebias: Learning to debias for recommendation. In *Proceedings of the International ACM SIGIR Conference on research and development in information retrieval* (pp. 21–30).

Chen, J., Dong, H., Wang, X., Feng, F., Wang, M., & He, X. (2021). Bias and debias in recommender system: A survey and future directions. *IEEE Transactions on Knowledge and Data Engineering*, 1–20.

Chen, J., Jiang, C., Wang, C., Zhou, S., Feng, Y., Chen, C., Ester, M., & He, X. (2021). CoSam: An efficient collaborative adaptive sampler for recommendation. *ACM Transactions on Information Systems*, *39*(3), 1–24.

Chen, J., Wang, C., Zhou, S., Shi, Q., Chen, J., Feng, Y., & Chen, C. (2020). Fast adaptively weighted matrix factorization for recommendation with implicit feedback. In *Proceedings of the AAAI Conference on artificial intelligence* (pp. 3470–3477).

Chen, J., Wang, C., Zhou, S., Shi, Q., Feng, Y., & Chen, C. (2019). Samwalker: Social recommendation with informative sampling strategy. In *The world wide web conference* (pp. 228–239).

Chen, C., Zhang, M., Ma, W., Liu, Y., & Ma, S. (2020). Jointly non-sampling learning for knowledge graph enhanced recommendation. In *Proceedings of the International ACM SIGIR Conference on research and development in information retrieval* (pp. 189–198).

Ding, J., Quan, Y., He, X., Li, Y., & Jin, D. (2019). Reinforced negative sampling for recommendation with exposure data. In *Proceedings of the international joint conference on artificial intelligence* (pp. 2230–2236).

Guo, X., Lin, W., Li, Y., Liu, Z., Yang, L., Zhao, S., & Zhu, Z. (2020). DKEN: Deep knowledge-enhanced network for recommender systems. *Information Sciences*, *540*, 263–277.

Guo, Z., & Wang, H. (2021). A deep graph neural network-based mechanism for social recommendations. *IEEE Transactions on Industrial Informatics*, *17*(4), 2776–2783.

Guo, Q., Zhuang, F., Qin, C., Zhu, H., Xie, X., Xiong, H., & He, Q. (2022). A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*, *34*(8), 3549–3568.

Hamilton, W. L., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in neural information processing systems* (pp. 1024–1034).

Hou, Y., Zhang, J., Cheng, J., Ma, K., Ma, R. T. B., Chen, H., & Yang, M.-C. (2020). Measuring and improving the use of graph information in graph neural networks. In *International conference on learning representations*.

Hu, B., Shi, C., Zhao, W. X., & Yu, P. S. (2018). Leveraging meta-path based context for top- n recommendation with a neural co-attention model. In *Proceedings of the ACM SIGKDD International conference on knowledge discovery and data mining* (pp. 1531–1540).

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *International conference on learning representations*.

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International conference on learning representations*.

Li, Q., Wang, X., Wang, Z., & Xu, G. (2022). Be causal: De-biasing social network confounding in recommendation. *ACM Transactions on Knowledge Discovery from Data*.

Lin, Y., Liu, Z., Sun, M., Liu, Y., & Zhu, X. (2015). Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 2181–2187).

Lin, Y., Xu, B., Feng, J., Lin, H., & Xu, K. (2021). Knowledge-enhanced recommendation using item embedding and path attention. *Knowledge-Based Systems*, *233*, Article 107484.

Lu, Y., Dong, R., & Smyth, B. (2018). Why I like it: multi-task learning for recommendation and explanation. In *Proceedings of the ACM conference on recommender systems* (pp. 4–12).

Palumbo, E., Monti, D., Rizzo, G., Troncy, R., & Baralis, E. (2020). Entity2rec: Property-specific knowledge graph embeddings for item recommendation. *Expert Systems with Applications*, *151*(8), Article 113235.

Saito, Y., Yaginuma, S., Nishino, Y., Sakata, H., & Nakata, K. (2020). Unbiased recommender learning from missing-not-at-random implicit feedback. In *The ACM International conference on web search and data mining* (pp. 501–509).

Sang, L., Xu, M., Qian, S., & Wu, X. (2021). Knowledge graph enhanced neural collaborative recommendation. *Expert Systems with Applications*, *164*, Article 113992.

Schlichtkrull, M., Kipf, T. N., Bloem, P., Berg, R. v. d., Titov, I., & Welling, M. (2018). Modeling relational data with graph convolutional networks. In *European semantic web conference* (pp. 593–607).

Son, L. H. (2016). Dealing with the new user cold-start problem in recommender systems: A comparative review. *Information Systems*, *58*, 87–104.

Sun, Z., Deng, Z., Nie, J., & Tang, J. (2019). Rotate: Knowledge graph embedding by relational rotation in complex space. In *International conference on learning representations*.

Sun, Z., Yu, D., Fang, H., Yang, J., Qu, X., Zhang, J., & Geng, C. (2020). Are we evaluating rigorously? benchmarking recommendation for reproducible evaluation and fair comparison. In *The ACM Conference on recommender systems* (pp. 23–32).

Tang, X., Wang, T., Yang, H., & Song, H. (2019). AKUPM: Attention-enhanced knowledge-aware user preference model for recommendation. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1891–1899).

Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., & Bouchard, G. (2016). Complex embeddings for simple link prediction. In *Proceedings of the international conference on machine learning* (pp. 2071–2080).

Wang, C., Chen, J., Zhou, S., Shi, Q., Feng, Y., & Chen, C. (2021). SamWalker++: recommendation with informative sampling strategy. *IEEE Transactions on Knowledge and Data Engineering*.

Wang, W., Feng, F., He, X., Zhang, H., & Chua, T.-S. (2021). Clicks can be cheating: Counterfactual recommendation for mitigating clickbait issue. In *Proceedings of the International ACM SIGIR conference on research and development in information retrieval* (pp. 1288–1297).

Wang, X., He, X., Cao, Y., Liu, M., & Chua, T. (2019). KGAT: knowledge graph attention network for recommendation. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 950–958).

Wang, X., He, X., Wang, M., Feng, F., & Chua, T. (2019). Neural graph collaborative filtering. In *Proceedings of the international ACM SIGIR conference on research and development in information retrieval* (pp. 165–174).

Wang, X., Huang, T., Wang, D., Yuan, Y., Liu, Z., He, X., & Chua, T.-S. (2021). Learning intents behind interactions with knowledge graph for recommendation. In *Proceedings of the web conference* (pp. 878–887).

Wang, F., Li, Y., Zhang, Y., & Wei, D. (2021). KLGCN: Knowledge graph-aware light graph convolutional network for recommender systems. *Expert Systems with Applications*, *195*, Article 116513.

Wang, Z., Lin, G., Tan, H., Chen, Q., & Liu, X. (2020). CKAN: collaborative knowledge-aware attentive network for recommender systems. In *Proceedings of the international ACM SIGIR conference on research and development in information retrieval* (pp. 219–228).

Wang, X., Xu, Y., He, X., Cao, Y., Wang, M., & Chua, T.-S. (2020). Reinforced negative sampling over knowledge graph for recommendation. In *The world wide web conference* (pp. 99–109).

Wang, H., Zhang, F., Wang, J., Zhao, M., Li, W., Xie, X., & Guo, M. (2018). Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *Proceedings of the ACM international conference on information and knowledge management* (pp. 417–426).

Wang, H., Zhang, F., Zhang, M., Leskovec, J., Zhao, M., Li, W., & Wang, Z. (2019). Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In *Proceedings of the ACM SIGKDD International conference on knowledge discovery and data mining* (pp. 968–977).

Wang, H., Zhang, F., Zhao, M., Li, W., Xie, X., & Guo, M. (2019). Multi-task feature learning for knowledge graph enhanced recommendation. In *The world wide web conference* (pp. 2000–2010).

Wang, H., Zhao, M., Xie, X., Li, W., & Guo, M. (2019). Knowledge graph convolutional networks for recommender systems. In *The world wide web conference* (pp. 3307–3313).

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2021). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(1), 4–24.

Wu, S., Sun, F., Zhang, W., Xie, X., & Cui, B. (2021). Graph neural networks in recommender systems: a survey. *ACM Computing Surveys*.

Xiao, C., Liu, C., Ma, Y., Li, Z., & Luo, X. (2020). Time sensitivity-based popularity prediction for online promotion on Twitter. *Information Sciences*, *525*, 82–92.

Yang, Z., & Dong, S. (2020). HAGERec: Hierarchical attention graph convolutional network incorporating knowledge graph for explainable recommendation. *Knowledge-Based Systems*, *204*(9), Article 106194.

Ye, M., Yin, P., Lee, W.-C., & Lee, D.-L. (2011). Exploiting geographical influence for collaborative point-of-interest recommendation. In *Proceeding of the international ACM SIGIR conference on research and development in information retrieval* (pp. 325–334).

Yin, R., Li, K., Zhang, G., & Lu, J. (2019). A deeper graph neural network for recommender systems. *Knowledge-Based Systems*, *185*(12), Article 105020.

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., & Leskovec, J. (2018). Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 974–983).

Yu, X., Ren, X., Sun, Y., Gu, Q., Sturt, B., Khandelwal, U., Norick, B., & Han, J. (2014). Personalized entity recommendation: a heterogeneous information network approach. In *ACM International conference on web search and data mining* (pp. 283–292).

Zhang, Z., Cai, J., Zhang, Y., & Wang, J. (2020). Learning hierarchy-aware knowledge graph embeddings for link prediction. In *The AAAI conference on artificial intelligence* (pp. 3065–3072).

Zhang, F., Yuan, N. J., Lian, D., Xie, X., & Ma, W. (2016). Collaborative knowledge base embedding for recommender systems. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 353–362).

Zhong, T., Zhang, S., Zhou, F., Zhang, K., Trajcevski, G., & Wu, J. (2020). Hybrid graph convolutional networks with multi-head attention for location recommendation. *World Wide Web*, *23*(6), 3125–3151.

Zhou, K., Zhao, W. X., Bian, S., Zhou, Y., Wen, J., & Yu, J. (2020). Improving conversational recommender systems via knowledge graph based semantic fusion. In *The ACM SIGKDD conference on knowledge discovery and data mining* (pp. 1006–1014).

Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., & Koutra, D. (2020). Beyond homophily in graph neural networks: Current limitations and effective designs. In *Advances in neural information processing systems* (pp. 7793–7804).