Knowledge-Based Systems 258 (2022) 110049

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

# Recommendations with residual connections and negative sampling based on knowledge graphs

Yuanyuan Liu [a], Zhaoqian Zhong [a], Chao Che [a,*], Yongjun Zhu [b]

[a] Key Laboratory of Advanced Design and Intelligent Computing Ministry of Education, Dalian University, Dalian, China
[b] Department of Library and Information Science, Yonsei University, Republic of Korea

ARTICLE INFO

*Article history:*
Received 7 May 2022
Received in revised form 14 October 2022
Accepted 15 October 2022
Available online 21 October 2022

*Keywords:*
Recommendation system
Knowledge graph
Graph neural networks
Negative sampling
Residual network
Gated mechanism

ABSTRACT

A knowledge graph (KG) contains a large amount of well-structured external triple information that can effectively solve the problems of poor interpretability in collaborative filtering. Recently, recommendation system (RS) models relying on graph neural networks (GNNs) have been widely developed, but the increase of GNN layers inevitably leads to over-smoothing problems. Meanwhile, most of the current KG-based negative sampling strategies randomly collect negative samples from unobserved data to train RS models. However, these strategies are insufficient to generate negative samples reflecting genuine user demands. To overcome these obstacles, we design a model called **K**nowledge **G**raph **R**esidual **N**egative **S**ampling Recommendation (KGRNS), which utilizes residual connections and pooling operation to alleviate the over-smoothing problem, and generate high-quality negative samples by negative sampling. Specifically, we devise residual connections on each output layer of the GNN and then utilize sum pooling operation to mitigate the effects of the over-smoothing problem on the model. In addition, to generate high-quality negative samples, we create a gated strategy to mix the knowledge of both positive and negative samples to generate synthetic negative samples and then select the virtual negative sample that is closest to the positive ones through a theoretically backed hard negative sample select strategy. We conducted broad experiments on three datasets. The experimental results showed that KGRNS performed considerable enhancements over state-of-the-art methods. Ablation studies validated the effectiveness of each part of the KGRNS.

© 2022 Elsevier B.V. All rights reserved.

## 1. Introduction

Information overload is a serious problem we must face when dealing with massive amounts of information in web services, such as E-commerce, social platforms, and online shopping. Recommendation systems (RSs) can essentially alleviate the knowledge overload issue and have become an irreplaceable part of almost all applications. The classical collaborative filtering (CF) [1–4] method has been studied and employed in recommendation systems to a wide extent, which imitates that users with much the same behaviors may have comparable preferences. However, these CF-based [5–7] approaches cannot utilize auxiliary knowledge (e.g., item characters and context) and inevitably have sparsity and cold-start problems.

A knowledge graph (KG) has more semantic connections between items than other auxiliary information, which might benefit the exploration of potential relationships between items and boost the precision of the recommendation outcomes. A KG connects users' historical preferences with the items to be recommended, thus enhancing the interpretability of the RSs. Therefore, KGs have a higher preference for improving the accuracy, diversity, and interpretability of recommendations.

To exploit KG information, researchers have developed many KG-based [8–14] recommendation systems, such as KGAT [9], CKAN [11], and KGIN [12]. The early methods utilized the knowledge graph embedding (KGE) [15] approach. For example, CKE [10] uses TransR to attain item representations for KGs and successfully embeds user and item representations into a low-dimensional vector space for prediction. However, CKE focuses on triple representations and cannot model higher-order information [16–18] to learn deeper potential user–item (U–I) interactions. Researchers have begun to utilize higher-order information and explore end-to-end methods [12,19,20] to model higher-order relationships with recursive propagation from embeddings of neighbor nodes (which can be users, items, or attributes). These approaches use multi-layer GNNs [21–28] to model higher-order item connection information to discover potential item attributes, taking advantage of the valuable knowledge contained in a KG. KGIN improves the recommendation performance by refining U–I interactions and modeling U–I representations.

* Corresponding author.
*E-mail addresses:* liuyuanyuan@s.dlu.edu.cn (Y. Liu), zhaoqianzhong@gmail.com (Z. Zhong), chechao@gmail.com (C. Che), zhu@yonsei.ac.kr (Y. Zhu).

https://doi.org/10.1016/j.knosys.2022.110049
0950-7051/© 2022 Elsevier B.V. All rights reserved.

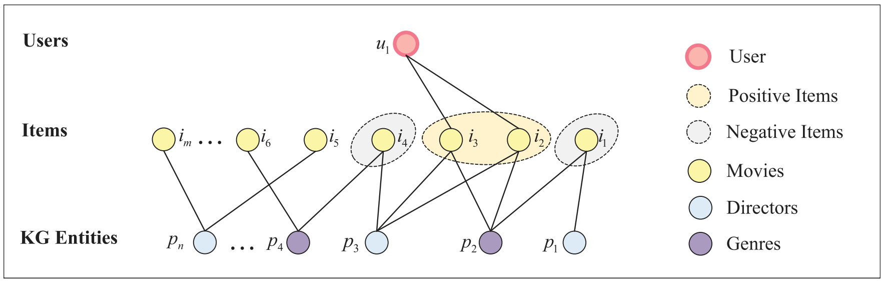


**Fig. 1.** Simple example of distilling negative samples from a knowledge graph (KG). Yellow circles indicate movies, light blue circles indicate the movie directors, and purple circles indicate the movie genres. Positive items indicate movies user $u_1$ has watched, negative items indicate movies that user $u_1$ has not seen so far. Having overlapping KG entities with $i_2$ and $i_3$, $i_1$ and $i_4$ are more likely to be of less interest to $u_1$.

Although all the abovementioned methods for modeling relations by using higher-order information have achieved good recommendations, there are still some unsolved problems:

(1) Higher-order relations cause the over-smoothing problem. Most of these methods employ multi-layer GNNs to model relationships, but over-smoothing becomes a concern as the number of GNN layers increases. The over-smoothing problem results in the GNN output having almost the same representation of each node. Although pooled GNN output layer representations can help alleviate the above issue to some extent, this method leads to insufficient differentiation of the final input node representation. As a result, the final prediction is unsatisfactory.

(2) The model does not distinguish potential positive samples from negative samples caused by an imperfect negative sampling strategy. Reasonable negative samples can assist the RS in learning the boundary of positive samples and negative samples. Most interactions in real-world scenarios are in the mode of implicit feedback, posing a primary opposition to recommendation model learning, for example, how to extract negative informative samples from numerous negative samples.

Consider the scenario in Fig. 1 as an example, in which user $u_1$ has watched movies $i_2$ and $i_3$, both directed by the same director $p_3$ and belonging to the same genre $p_2$. We may deduce that $u_1'$s interest is influenced by the combination of director $p_3$ and genre $p_2$. Therefore, $u_1$ is more probably to know different movies (e.g., $i_4$) directed by $p_3$ with other genres but is barely interested in these movies. Similarly, as for negative sampling, $i_4$ can provide high-quality negative knowledge. The presented KG-based CF recommendation methods solely use unobserved items as negative samples for uniform distribution model training. These random sampling strategies cannot generate high-quality negative samples that accurately reflect the actual needs of users. In contrast, these strategies may sample items that interact with users but may not be of interest to users, thus polluting the recommendation performance.

In this paper, we propose the **K**nowledge **G**raph **R**esidual **N**egative **S**ampling Recommendation (KGRNS) to solve the abovementioned challenges. We use residual networks [29,30] and sum pooling operation to solve the over-smoothing problem caused by higher-order relations in GNNs. First, we add the previous layer's output representation to the current layer to certify the diversity of the output knowledge. And then, we perform sum pooling operation to pool the knowledge from all the output layers to make the nodes with higher-order information more discriminative to alleviate the over-smoothing problem. To generate high-quality negative samples, we use a gated mechanism to inject the knowledge of positive samples into the knowledge of negative samples and adaptively learn the quantity of the injected information to pollute the negative samples. This paper uses a hard negative sample selection strategy to produce virtual negative samples. We select the virtual negative samples closest to the positive ones so that the recommender can train on more complex data and enhance the generalization ability of the KGRNS model.

The main contributions of this work are as follows:

- We emphasize the significant effects of using our proposed negative sampling strategy, residual connection strategy, and sum pooling operation on the recommendation model.
- We design a model called KGRSN, which improves the RSs' accuracy, diversity, and interpretability under the framework of GNNs and alleviates the over-smoothing problem.
- We performed extensive experiments on three public benchmark datasets. The outcomes demonstrated that our proposed KGRNS consistently and notably outperformed state-of-the-art models in recommendation performance.

## 2. Related work

### 2.1. Negative sampling

Negative sampling [31–34] is the process of constructing negative samples opposite to positive samples through a certain strategy. Without negative sampling, optimizing all negative samples would result in a computationally inefficient model. Assuming the user's preferred items have higher scores than other items, we can obtain an order list of unobserved items. Nevertheless, the ranked item list for each user is usually derived only from implicit feedback, which contains only positive observations. A solution is to assume that users like the observed items rather than unobserved ones. Negative sampling methods commonly used in recommendation systems are static negative sampling [31], hard negative sampling [35], adversarial sampling [32], graph-based sampling [33,34,36], and additional-data-enhanced sampling [37]. In this study, graph-based negative sampling was used. KGPolicy [33] utilizes auxiliary information in KG and reinforcement learning methods to find high-quality negative samples. PinSage [36] proposes strong negative sampling based on PageRank scores, designed to further exploit the structural information in a KG.

### 2.2. Graph neural networks

GNNs [9,11,20,38] generalize deep neural networks to graph-structured data and have performed the most advanced performances in several tasks, such as node classification [27,39], link prediction [40,41], and node clustering [42,43]. The basic thought of GNNs is to aggregate the features of node neighbors by using a neural network. GNNs can capture both the graph structure and the properties of individual nodes, making it more suitable for downstream tasks than traditional graph embedding models, especially recommendation tasks. For example, NGCF [38] uses GNNs in U–I bipartite graphs for CF. KGAT [9], CKAN [11], and KGCN [20] extend items to KGs and use GNNs to improve the representations of items. Many deep learning models applied to various tasks [30,44–49] based on hypergraph neural networks (HGNN) [50] have been proposed recently, such as HCNH [45], HCoN [46], and MedRec [49]. The edges on a hypergraph are called hyperedges. A hyperedge can connect multiple nodes, while an edge on a traditional graph can only connect two nodes. Therefore, HGNN can represent complex higher-order relationships between nodes better than GNN. However, the main goal of our work is to alleviate the over-smoothing problem caused by multi-layer GNNs; therefore, in this article, we only worked on GNNs, and we will further investigate hypergraph-based recommendation models in future work.

### 2.3. GNN-based recommendation methods

GNN-Based Recommendation Methods [9,11,19,51] are based on the information aggregation mechanism of GNNs. Typically, these methods use the information represented by the ego node updated from one-hop nodes. When this propagation was performed recursively, the information of high-order entities could be encoded in the representation. As a result, these methods based on GNN can simulate higher-hop connections. For example, KGAT [9] combines U–I interactions and KGs into a heterogeneous graph. However, as far as we know, most of the current GNN-based approaches do not utilize high-quality negative sampling strategies [9,12,37].

The approach in our work differs from these previous approaches in terms of negative sampling. We focus on employing a negative sampling strategy to generate high-quality hard negative samples to train recommendation models after GNN information aggregation to strengthen the generalization ability of the recommendation models.

## 3. Task formulation

***Definitions.*** The input of a recommendation system usually includes a **user set** $U = \{u\}$ and an **item set** $I = \{i\}$. We concentrate on the implicit feedback [52] in RS, where the preference singles are implicit (e.g., view and click) about users.

Implicit feedback of RS is denoted as $O^+ = \{(u, i) \mid u \in U, i \in I\}$, where each $(u, i)$ pair denotes an interaction between user $u$ and observed positive item $i$. The U–I bipartite graph is a storage form of U–I interactions, we define the graph as $G_1 = \{u, i, y_{ui} \mid u \in U, i \in I\}$, where the interaction between user $u$ and item $i$ is denoted as $y_{ui}$. The link $y_{ui} = 1$ means the interaction between user $u$ and item $i$ has been observed; otherwise, $y_{ui} = 0$. Notably, $y_{ui} = 0$ does not mean that user $u$ is not interested in item $i$, it may also indicate user $u$ is potentially interested but not knowing about it. KG is a structured knowledge base, whose basic component unit is the "head entity-relationship-tail entity" triad, and the representation of a KG is based on amounts of triad information. We extract the initial vector representation $e_i^{(0)}$ of the items from the knowledge graph $G_2 = \{(h, r, t) \mid h, t \in \varepsilon, r \in \mathcal{R}\}$, where $\varepsilon$, $\mathcal{R}$, $h$, and $t$ represents the set of the entities, relations, head entity, and tail entity, respectively.

***Task description.*** The sources of the GNN inputs are the U–I bipartite graph $G_1$ ($U{-}I$ interaction graph shown in Fig. 2) and the KG $G_2$ (KG graph shown in Fig. 2). After establishing user behavior and item knowledge, the rich information in $G_1$ and $G_2$ guide sampler learning. Our goal is to generate high-quality negative samples. The recommendation model learns a prediction function $\hat{y}_{ui} = F(u, i \mid G_1, G_2, \Theta)$ motivated by these samples, and the function can predict potential interest in negative items, where $\hat{y}_{ui}$ denotes the probability of user $u$ interacting with item $i$.

## 4. Methodology

In this section, we introduce our presented KGRNS, which is illustrated in Fig. 2. KGRNS consists of five main parts: (1) embedding layer, which models each entity as an embedding vector; (2) residual connection, which alleviates the over-smoothing problem by adding residual connections between the inputs and outputs of the GNN layers; (3) negative sampling, which selects high-quality synthetic negative samples by a hard negative selection strategy; (4) embedding combination, which further alleviates the over-smoothing problem through sum-based pooling; (5) prediction layer, which calculates the score of user embedding and item embedding by an inner product and selects the item with the highest score to the user.

### 4.1. User embedding and item embedding in GNN propagation

The U–I bipartite graph and KG enables the GNN to generate user and item embeddings with high expressiveness for RSs. In web-scale recommendation systems, user and item IDs are usually encoded as one-hot vectors. Since user $u$ is initially calculated in the same way as item $i$, we use $e_h$ to refer collectively to $e_u$ and $e_i$. We first transform the user ID and item ID into the embeddings $e_h = E \cdot x_h$, where $E \in \mathcal{R}^{(M+N)\times d_0}$ is the initial embedding table for users and items, $x_h$ is the ID (one-hot vector) of a user or an item, $e_h \in \mathcal{R}^{d_0}$ is the gathered embedding, $M$ and $N$ are the numbers of users and items, respectively, and $d_0$ is the dimension of the initial embedding.

#### 4.1.1. User embedding

Following the KGIN approach [12], we use finer-grained intent to model user embeddings to symbolize the interactions between users and items. The weight of each neighbor in the propagation process is produced by an attention mechanism [12,28]. Various amounts of intent information are combined for different users to guarantee that different intents lead to different contributions to user representations. Intents improve the quality of the user embedding and emphasize the relevance of higher-order connections by allowing for better learning of user preferences. $e_p$ is the embedding of intent $p$, which is defined as follows:

$$\mathbf{e_p} = \sum_{r \in \mathcal{R}} \delta(r, p)\mathbf{e}_r \tag{1}$$

where $e_r$ is the embedding of relation $r$, $\delta(r, p)$ is an attention score to quantify the importance of $e_r$, which is formulated as follows:

$$\delta(r, p) = \frac{\exp\left(w_{rp}\right)}{\sum_{r' \in \mathcal{R}} \exp\left(w_{r'p}\right)} \tag{2}$$

where $w_{rp}$ is a trainable weight specific to relation $r$ and intent $p$.

Furthermore, recent research [53] has shown that using feature transformations and nonlinear transformations in traditional GNN models does not improve the recommendation performance, and thus, we remove these two redundant operations. We aggregate the intention knowledge from historical items and generate embedding of user $u$ as follows:

$$\mathbf{e}_u^{(1)} = \frac{1}{|S_u|} \sum_{(p,i) \in S_u} \alpha(u, p)\mathbf{e}_p \odot \mathbf{e}_i^{(0)} \tag{3}$$

in which $e_u^{(1)} \in \mathcal{R}^d$ is the first-order embedding formulation of user $u$, $e_i^{(0)}$ is the initial ID embedding representation of item $i$, $\odot$ indicates the element-wise product, and $S_u$ represents the set of first-order connected entities around user $u$. For specific users,

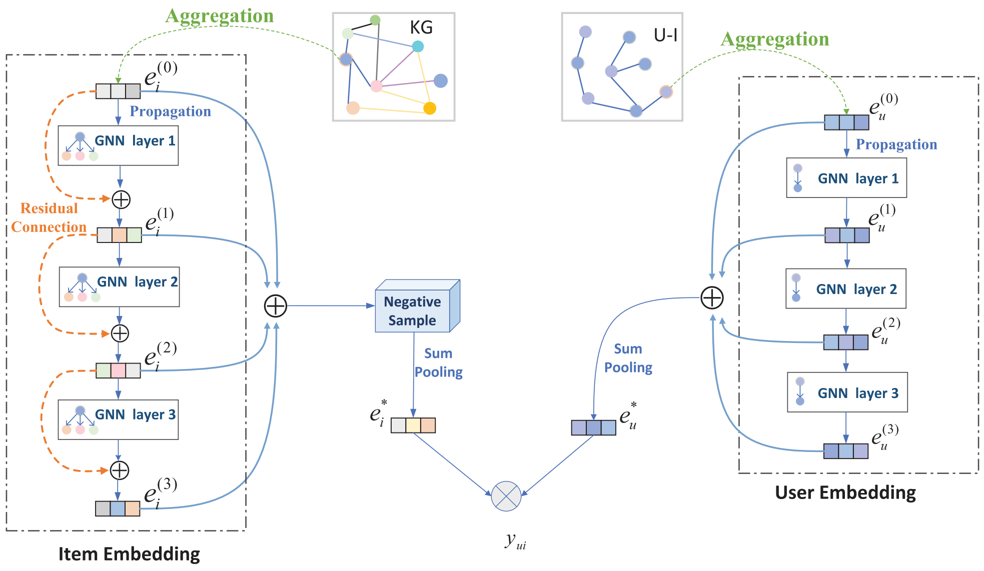


**Fig. 2.** Overall structure of Knowledge Graph Residual Negative Sampling Recommendation (KGRNS). $e^{(l)}$ denotes the $l$th layer embedding of node $e$. It takes the initial embeddings of users and items as inputs, then predicts the preferences of users to items. $\otimes$ indicates an inner product of the user embedding and the item embedding.

different intents have different impacts to motivating their behaviors. Therefore, we adopt $\alpha(u,p)$ to distinguish the importance of intent $p$ and define it as an attention score, which is formulated as follows:

$$\alpha(u,p)=\frac{\exp\left(\mathbf{e}_p^{\top}\mathbf{e}_u^{(0)}\right)}{\sum_{p'\in\mathcal{P}}\exp\left(\mathbf{e}_{p'}^{\top}\mathbf{e}_u^{(0)}\right)} \tag{4}$$

in which $e_u^{(0)}\in\mathcal{R}^d$ is the initial ID embedding representation of a given user $u$.

### 4.1.2. Item embedding

A KG contains a large amount of external item information, such as (*The Revenant*, *director*, *Alejandro Gonzalez Inarritu*), (*The Revenant*, *star*, *Leonardo DiCaprio*), and other triples contain substantial amounts of information about *The Revenant*, which can be generated by aggregating this information in the item representation. The useful information in the triples contains entities, attributes, and equally important relationships. However, the original GNN model cannot represent relationships, including critical semantic information. When aggregating information about item neighbors, using relationship information can help to describe the distinctions between various entities. We integrate relationship information from neighbor entities to generate an embedding representation of item $i$:

$$\mathbf{e}_i^{(1)}=\frac{1}{|S_i|}\sum_{(r,v)\in S_i}\beta(i,r)\mathbf{e}_r\odot\mathbf{e}_v^{(0)} \tag{5}$$

where $e_i^{(1)}$ is the embedding representation of item $i$ generated by the first GNN layer, $e_v^{(0)}$ represents the initial ID embedding of entity $v$, $S_i$ is the entities' neighbor set of item $i$ on first GNN layer, and $e_r$ is the embedding representation of relation $r$. $\beta(i,r)$ represents the attention score that distinguishes the importance of different relationships, which is expressed as follows:

$$\beta(i,r)=\frac{\exp\left(\mathbf{e}_r^{\top}\mathbf{e}_i^{(0)}\right)}{\sum_{r'\in\mathcal{R}}\exp\left(\mathbf{e}_{r'}^{\top}\mathbf{e}_i^{(0)}\right)} \tag{6}$$

For each triple $(i,r,v)$, we reveal the different information carried by the triples, even if they obtain the same entities.

## 4.2. Residual connection

Data smoothing [54] helps decrease the sparsity of CF as the number of stacked layers increases, but modeling higher-order relationships with a multi-layer GNN inevitably suffers from over-smoothing problems, which causes the output of the final representation layer of the GNN to converge. The over-smoothing problem makes it impossible to discriminate between various nodes and ignores the uniqueness of each user.

To solve the over-smoothing problem described above, we apply a graph residual learning architecture inspired by residual networks [29] to learn the item representation for that layer using the following residual representation function:

$$e_i^{(l)}=\frac{1}{|N_i|}\sum_{(r,v)\in N_i}\beta(i,r)e_r\odot e_v^{(l-1)}+\gamma e_i^{(l-1)} \tag{7}$$

where $e_i^{(l)}$ denotes the $l$th layer representation of item $i$, and $\gamma$ denotes an identity matrix for assigning weights to residual connections.

As the amount of GNN layers increases, lower-order features' influence becomes less dominant. By adding residual connections between the inputs and outputs of the layers, the features of the lower-order receptive fields can be rescaled to enhance the flow of knowledge in the network, thus alleviating the over-smoothing problem.

## 4.3. Negative sampling

For implicit feedback, the data is not explicitly labeled. To help the model better use this unlabeled data, we assume that the items that users have interacted with are positive, and we utilize a sampling strategy to label a portion of the items that users have not interacted with as negative. The process of selecting negative samples is called negative sampling.

Current KG-based recommendation systems use uniformly distributed random sampling of user interaction items [1,53,55] to select the negative examples, but the performance is not as effective as it could be (as mentioned in Section 1). We perform negative sampling by synthesizing virtual hard negative samples containing gated mixing and hierarchical sampling. Fig. 3 shows the detailed design of the negative sampling.

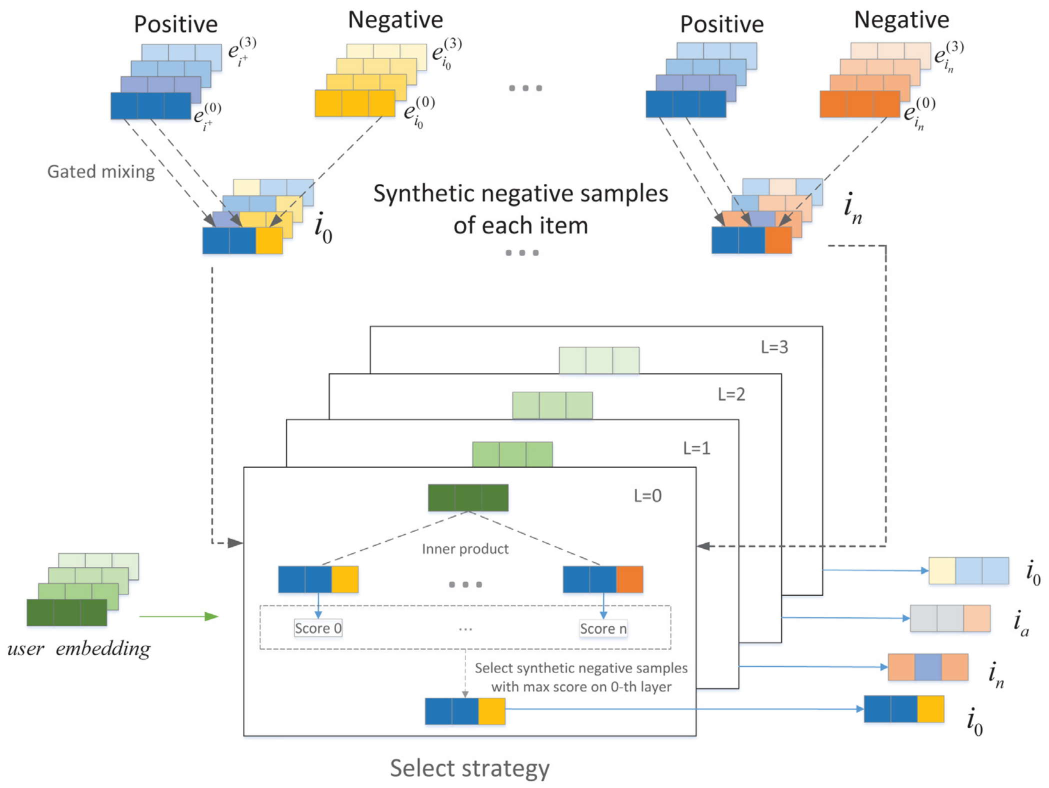


**Fig. 3.** Detailed design of the negative sample section. $e^{(0)}$ denotes the initial embedding of node $e$. $e^{(l)}$ denotes the $l$th graph neural network (GNN) layer embedding of node $e$. $e'^{(l)}$ denotes the $l$th layer embedding of the synthetic negative samples generated by gated mixing. We select synthetic negative samples by a selection strategy to predict which items the user would like in the future.

### 4.3.1. *Gated mixing mechanism in negative sampling*

Instead of using uniform negative sampling strategies, we utilize data augmentation and a metric learning method [56–58] to create negative samples with more negative information. First, for each $(u, i^+)$ pair, $n$ negative items are randomly selected by traditional negative sampling to form a set of candidate negative items $\varepsilon$ of size $n*(L+1)$. Each positive item in $(u, i^+)$ has $L+1$ vectors after $L$ layer GNNs. And each candidate negative sample $n \in \varepsilon$ contains $L+1$ vector representations of the output of the $L$-layer GNN. Motivated by the negative sampling method MixGCF [34], we add the positive sample information $e_{i^+}^{(l)}$ generated by each GNN layer with residual connection to the candidate negative sample set $\varepsilon$ to form a virtual candidate negative sample set $\varepsilon' = \left\{e_{v_n}'^{(l)}\right\}$, which $v_n$ denotes the $n$th sample. For each $(u, i^+)$ pair, a positive sample is mixed with $n$ negative samples in the gated mixing phase, respectively.

We exploit a nonlinear gate defined by Eqs. (6) and (7) by dimensional re-weighting to adapt the mixing ratio of positive and negative samples with feature granularity. The model learns the mixing ratio adaptively through a properly designed gated mechanism. For each candidate negative embedding $e_{i_n}^{(l)} \in \varepsilon$, the positive mixing operation is formalized as

$$\mathrm{e}_{i_n}'^{(l)} = \varphi^{(l)}\mathbf{e}_{i^+}^{(l)} + \left(1-\varphi^{(l)}\right)\mathrm{e}_{i_n}^{(l)}, \varphi^{(l)} \in (0,1) \tag{8}$$

where $\varphi^{(l)}$ is a mixing factor sampled from $(0, 1)$ and is formulated as

$$\varphi^{(l)} = \sigma\left(W^{(l)}e_{i_n}^{(l)} + b^{(l)}\right) \tag{9}$$

in which $W$ is a trainable weight, and $b$ is the bias.

By injecting positive sample information into the negative sample information via a gating mechanism, we can make the model learn the boundaries of the positive samples and negative samples more clearly, thus further improving the generalization ability of the KGRNS model.

### 4.3.2. *Stratification sampling in negative sampling*

In this subsection, we use the virtual candidate negative sample set $\varepsilon'$ obtained by gated mixing as the basis. We utilize GNN hierarchical aggregation sampling to obtain the final negative sample set $\varepsilon^{\sim}$ from virtual candidate negative sample set $\varepsilon'$. Specifically, for the GNN layer $l = 0$, we use Eq. (8) to obtain the scores by inner product $e_u^{(0)}$ with $e_{i_0}'^{(0)}, e_{i_1}'^{(0)}, \ldots, e_{i_n}'^{(0)}$ of virtual candidate negative sample set $\varepsilon'$ separately, and select the corresponding item with the largest score to be included in the final negative sample set $\varepsilon^{\sim}$. Larger scores indicate a closer representation of positive and negative samples. Similarly, the negative samples with the largest scores in each GNN layer are included in the final negative sample set $\varepsilon^{\sim}$. When specific to each layer $l$ $(0 \le l \le L)$, we choose the candidate negative sample embeddings $e_{i_x}'^{(l)}$ from $\varepsilon'^{(l)}$, which contains the embeddings of all layers of the candidate negative sample set $\varepsilon$, denoted as

$$\mathbf{e}_{i_x}'^{(l)} = \underset{\mathbf{e}_{i_n}'^{(l)} \in \varepsilon(l)}{\arg\max} f_Q(u,l) \cdot \mathbf{e}_{i_n}'^{(l)} \tag{10}$$

where $\cdot$ presents the inner product operation, $f_Q(u, l)$ is the query mapping that returns the embedding correlation with the $l$th hop of the target user $u$. A negative sample with the highest score from $n$ vector representations is selected as the sampling result of each GNN layer. Taking $L = 1$ as an example, we can take a sample $e_{i_a}'^{(0)}, e_{i_b}'^{(1)}$ from $\varepsilon'$. Note that $a$ and $b$ do not need to be distinguished because $a$ and $b$ may be the same item.

With the above steps, we can obtain a negative sample set $\varepsilon^{\sim}$ that consists of synthetic virtual negative samples. Since the synthetic virtual negative samples are closer to the positive samples in the vector representation space (shown in Fig. 5) than the real negative samples obtained by random sampling, it allows the model to recognize the boundary between positive and negative samples better.

#### 4.3.3. Loss of negative sampling

Due to the large scale of unobserved items (commonly in $O\left(|I|^2\right)$), the learning objective is usually simplified as the negative sampling loss:

$$\mathcal{L}_{NegS} = \max \prod_{i^+, i^- \sim f_s(u)} P_u\left(i^+ > i^- \mid \Theta\right) \tag{11}$$

where $i^+$ and $i^-$ denote the positive items and negative items, respectively, $P_u\left(a > b\right)$ represents a probability that user $u$ prefers item $b$ over $a$, $f_s(u)$ denotes the distribution of the negative sampling, and $\Theta$ is the parameter of the model. Current recommendation methods majorly select negative samples from a uniform distribution ($f_s(u)$ as $f_{uniform}(u)$) [1,52,53,55].

### 4.4. Embedding combination

Usually, GNNs are used for tasks such as KG link prediction or entity alignment, only using the last layer of the GNN output as the final output. In this paper, we make the embedding information richer and further alleviate the over-smoothing problem by combining the embeddings of all layers (including the initial embedding) as the final feature representation of each node, which is represented as

$$\mathbf{e}_u^* = f_{pool}\left(\mathbf{e}_u^{(0)}, \ldots, \mathbf{e}_u^{(L)}\right) \tag{12}$$

$$\mathbf{e}_i^* = f_{pool}\left(\mathbf{e}_{i_x}^{\prime(0)}, \ldots, \mathbf{e}_{i_y}^{\prime(L)}\right) \tag{13}$$

where $f_{pool}\left(\cdot\right)$ indicates that the application uses the same pooling operation in the current GNN-based recommender. A sum-pooling operation is used in this paper, expressed as

$$e_u^* = \sum_{s=0}^{L} e_u^{(s)} \tag{14}$$

$$e_i^* = \sum_{t=0}^{L} e_{i_x}^{\prime(t)} \tag{15}$$

We obtain the final user and item embedding representations by pooling operation and use them for recommendation prediction. We generate high-quality negative samples indicative of user preferences by constructing virtual negative samples. We also succeed at reducing the potential of collecting samples that are of interest to users but that they do not interact.

### 4.5. Model prediction and optimization

After executing the embedding combination defined by Eqs. (12) and (13), we obtain the final predicted representation $e_u^*$ and the representation $e_i^*$ used for prediction. Then, we predict their scores by executing an inner product of the user and item representations, and the calculation method is as follows:

$$\hat{y}_{ui} = e_u^{*T} e_i^* \tag{16}$$

We choose the paired Bayesian personalized ranking (BPR) loss [52] to remodel the historic interacted data, which supposes that the predicted scores for user of its observed items should be higher than those of not interacted items, expressed as

$$\mathcal{L}_{BPR} = \sum_{(u,i^+,i^-)\in O} -\ln\sigma\left(\hat{y}_{ui^+} - \hat{y}_{ui^-}\right) \tag{17}$$

where $O = \left\{\left(u, i^+, i^-\right) \mid \left(u, i^+\right) \in O^+, \left(u, i^-\right) \in O^-\right\}$, $O^+$ denotes the observed datasets and $O^-$ denotes the unobserved datasets, and $\sigma\left(\cdot\right)$ indicates the sigmoid function. The GNN networks update the model parameters by minimizing the following objective function:

$$\mathcal{L}_{KGRNS} = \mathcal{L}_{NegS} + \mathcal{L}_{BPR} + \lambda\|\Theta\|_2^2 \tag{18}$$

where $\Theta = \left\{e_u, e_i, e_p, e_r, w, b \mid u \in U, i \in I\right\}$ is the parameter set of the KGRNS model. To effectively avoid overfitting, we added the $L_2$ regularization, where $\lambda$ is the regularization factor.

### 4.6. Time complexity

The time cost of KGRNS mainly comes from aggregation scheme and negative sampling. In the aggregation stage, the time cost of updating entity embedding representations is $O(L|G|d)$, where $L$ is the number of GNN layers, $G$ is the number of triplets, and $d$ is the embedding size. In the negative sampling phase, the time cost is $O(L|N|d)$, where $N$ is the size of the negative candidate set. In total, the time complexity of KGRNS is $O(L|G|d + L|N|d)$.

## 5. Experiments

### 5.1. Experimental settings

#### 5.1.1. Dataset description

We used three datasets to evaluate our model: the Amazon-Book, Last-FM dataset published by KGAT [9], and the Alibaba-iFashion dataset released by KGIN [12]. In our experiments, we only listed the number of standard relationships and constructed the triples with inverse relationships in Table 1. The KG of Alibaba-iFashion had more first-order connections than the other two datasets.

Following prior studies [9,12,59], we used the aforementioned data processing methods. During the training stage, each interacted U–I was a positive sample, whereas items the user had not previously adopted were randomly selected as negative samples.

#### 5.1.2. Parameter settings

We implemented our KGRNS model in PyTorch. In fairness, we fixed the scale of the ID embeddings $d$ to 64, the optimizer we used was Adam [60]. For all methods, we set the batch size to 1024. The learning rate was tuned from $\left\{10^{-5}, 10^{-4}, 10^{-3}\right\}$, the dynamic weights of auxiliary constraints (e.g., $L_2$ regularization) were searched for in the set $\left\{10^{-5}, 10^{-4}, \ldots, 10^{-1}\right\}$. For RS models based on GNN, the amount of GNN layers $L$ was chosen from $\{1, 2, 3\}$. In the candidate negative item set $\varepsilon$, the number of items $n$ had the values of 64, 64, and 32 in Amazon-Book, Last-FM, and Alibaba-iFashion, respectively. We use the original scale of connected items and batch size of KGNN-LS and CKAN. We initialized the model parameters with Xavier [61].

Considering it is time-consuming to do sensitivity analysis on all datasets, we only did experiments on Alibaba-iFashion dataset about the number of negative samples $n$. To ensure the quality of the candidate negative sample set, we set the minimum sampling value to 32, the results were shown in Table 2.

We denoted the RS model with three GNN aggregation layers as KGRNS-3 and used a similar notation for the others. Moreover, in Section 5.3.2, we observe their influence by varying $L$ in $\{1, 2, 3\}$.

**Table 1**
Statistics of the datasets.

| | | Amazon-book | Last-FM | Alibaba-iFashion |
|---|---|---|---|---|
| User–item interaction | Users | 70,679 | 23,566 | 114,737 |
| | Items | 24,915 | 48,123 | 30,040 |
| | Interactions | 847,733 | 3,034,796 | 1,781,093 |
| Knowledge graph | Entities | 88,572 | 58,266 | 59,156 |
| | Relations | 39 | 9 | 51 |
| | Triples | 2,557,746 | 464,567 | 279,155 |

**Table 2**
Hyperparameter sensitivity analysis on Alibaba-iFashion about negative sample number *n*.

| Model | *n* | Alibaba-iFashion | | | |
|---|---|---|---|---|---|
| | | *recall@20* | *ndcg@20* | *precision@20* | *hit rate@20* |
| KGRNS | 32 | **0.1302** | **0.0832** | **0.0220** | **0.3544** |
| | 64 | 0.1246 | 0.0804 | 0.0211 | 0.3406 |
| | 128 | 0.1245 | 0.0803 | 0.0210 | 0.3403 |

### 5.1.3. Alternative baselines

We compared the proposed KGRNS with state-of-the-art methods, including MF, CKE, and KGAT, KGNN-LS, CKAN, RGCN and KGIN.

- MF [2] deliberates the U–I contacts and not unconnected KGs, using users' ID embeddings and items' ID embeddings to predict.
- CKE [10] is an embedding-based approach that utilizes the KG embedding of TransR-derived [28] entities as the ID embedding of items in the MF framework.
- KGNN-LS [19] is an RS model based on GNN, that generates user-specific item representations by considering user preferences for KG relationships and label smoothing in the information aggregation phase.
- KGAT [9] is a GNN-based recommendation model. It generates user embeddings and item embeddings by employing aggregation operation. Relationships are used as attention weights in the adjacency matrix.
- CKAN [11] is built based on KGNN-LS, which obtains user and item representations with different aggregation strategies on KG.
- R-GCN [62] combines connected nodes by treating several KG relations as different information flow channels.
- KGIN [12] adopts U–I relationships with finer-grained intent and utilizes long-term information of relational paths, revealing the user intent behind the interactions for the model capacity and interpretability.

### 5.1.4. Evaluation metrics

In the evaluation stage, we implemented the all-ranking strategy [63]. To determine the effectiveness of the top-$K$ RS, we adopted the widely adopted protocols [1,63,64] *recall@K* and *ndcg@K*, where $K$ was set by default to 20. *recall* reflected the proportion of correlated outcomes that were correctly predicted as a percentage of all correlated outcomes, while *ndcg* (Normalized Discounted Cumulative Gain) was used to estimate the accuracy of the ranking outcomes. To further validate the recommendation accuracy and diversity, we added two new evaluation metrics: *precision@K* and *hit rate@K*. Where *precision* denoted the percentage of items in the top-$K$ recommended list that user has interacted with and *hit rate* denoted whether the top-$K$ recommended list contains the item that user clicked on. We describe comparative experiments with different $K$ values for four evaluation metrics above-mentioned in Section 5.2.2. We present the average metrics in the test set for all users.

**Table 3**
Overall performance comparison. *%Imp.* denotes the relative improvements of the best performing method (bolded) over the strongest baselines (underlined).

| | Amazon-book | | Last-FM | | Alibaba-iFashion | |
|---|---|---|---|---|---|---|
| | *recall* | *ndcg* | *recall* | *ndcg* | *recall* | *ndcg* |
| MF | 0.1300 | 0.0678 | 0.0724 | 0.0617 | 0.1095 | 0.0670 |
| CKE | 0.1342 | 0.0698 | 0.0732 | 0.0630 | 0.1103 | 0.0676 |
| KGAT | 0.1487 | 0.0799 | 0.0873 | 0.0744 | 0.1030 | 0.0627 |
| KGNN-LS | 0.1362 | 0.0560 | 0.0880 | 0.0642 | 0.1039 | 0.0557 |
| CKAN | 0.1442 | 0.0698 | 0.0812 | 0.0660 | 0.0970 | 0.0509 |
| R-GCN | 0.1220 | 0.0646 | 0.0743 | 0.0631 | 0.0860 | 0.0515 |
| KGIN | <u>0.1687</u> | <u>0.0915</u> | <u>0.0978</u> | <u>0.0848</u> | <u>0.1147</u> | <u>0.0716</u> |
| KGRNS-1 | 0.1795 | 0.0980 | 0.1047 | 0.0932 | 0.1271 | 0.0817 |
| KGRNS-3 | **0.1820** | **0.1006** | **0.1117** | **0.1017** | **0.1302** | **0.0832** |
| *%Imp.* | 7.88% | 9.94% | 14.21% | 19.92% | 13.51% | 16.20% |

## 5.2. Superiority of KGRNS

### 5.2.1. Accuracy comparison

We start with the analysis with respect to *recall*@20 and *ndcg*@20. The results are presented in Table 3. We observed the following:

- KGRNS consistently outperformed all the baselines across the three datasets. Specifically, it achieved significant improvements over the most powerful baselines with respect to *ndcg*@20 by 9.94%, 19.92%, and 16.20% for Amazon-Book, Last-FM, and Alibaba-iFashion, respectively. Owing to the negative sampling strategy and relational modeling, KGRNS can better identify the boundaries between positive and negative samples, and the experimental results in Table 3 illustrate the effectiveness of KGRNS:
  (1) By using the interpolation operation with gating, the pooled positive sample embeddings were mixed with the initial negative sample embeddings by a certain ratio to generate the candidate negative samples.
  (2) Through the mixing operation with the hard negative sample selection strategy, inner product of candidate negative samples of each layer with the user embeddings were computed, and the negative sample with the highest score of each layer was selected as the final hard negative sample representation of each layer.
  (3) Due to our relational path aggregation scheme, compared with GNN-based baselines (such as KGAT, CKAN, and KGNN-LS), KGRNS could retain comprehensive information on all paths.
  (4) Applying various aggregation strategies on the U–I interaction graph $G_1$ and the knowledge graph $G_2$ enables the KGRNS to better represent the collaboration information into the user and item embeddings.
- By analyzing the performance of the KGRNS on three datasets, we found that the improvement for Alibaba-iFashion was more significant than that of Amazon-Book. It is reasonable because of the following:

(1) Compared to Alibaba-iFashion, the interactions on Amazon-Book provided denser and richer information, which could increase the complexity of its deep GNN training and lead to an over-smoothing problem.
(2) First-order connectivity dominated the KG triples in Alibaba-iFashion. It demonstrated that by constructing virtual hard negative samples, the KGRNS improved the ability of the model to distinguish between positive and negative samples and could be more effective, especially for shallow-interaction sparse datasets.

- The GNN-based methods were superior to CKE on Amazon-Book and Last-FM, indicating the significance of long-range connection modeling. However, the CKE outperformed them on Alibaba-iFashion. One of the possible reasons was that the GNN-based methods involved additional nonlinear feature transformations, which were quite burdensome for training and degraded the performance [53].
- KGIN had the best performance on all three data sets and was the strongest baseline in Table 3. In addition, KGRNS with only one GNN layer outperformed the KGIN model on all three datasets. The results reconfirmed the importance of our proposed negative sampling method and residual connectivity for recommendations.

### 5.2.2. Accuracy-diversity comparison

Most of the current GNN-based recommendation models [9,12,33,65,66] focus on *recall*@20 and *ndcg*@20 evaluation metrics, and focus on recommending popular items to users frequently, but this may also lead to a decrease in users' interest to this recommendation system. Therefore, it is important to add diverse evaluation metrics. We compared the state-of-the-art recommendation model KGIN with our proposed KGRNS model on Alibaba-iFashion dataset using four evaluation metrics, *recall@K*, *ndcg@K*, *precision@K* and *hit rate@K*. We set the value of $K$ to 10, 20, 30, 50, and 100. The experimental results were shown in Fig. 4. Our KGRNS model outperformed the KGIN model in all four metrics with different $K$ values, which again verified that KGRNS performed superiorly in terms of both recommendation accuracy and diversity.

## 5.3. Validation of individual parts of KGRNS

As negative sampling is the core of KGRNS, we focused on the negative sampling component and conducted ablation studies to examine the KGRNS effectiveness. We compared the effects of negative sampling and residual modules, the interpolation operation method in the negative sampling, and the number of GNN aggregation layers.

### 5.3.1. Effect of negative sampling operation, residual operation and sum-pooling operation

We designed ablation experiments w/o Neg (without Negative sampler), w/o Res (without Residual networks), and w/o Pool (without the final pooling of all the embedding layers) to confirm the effectiveness of each model separately, and Table 4 summarized the experimental results.

Removing the negative sampling module reduced the prediction accuracy greatly compared to that of KGRNS, as shown in Table 4, demonstrating the significance of negative sampling. At the same time, we reduced the dimensionality of the generated vectors and visualized them in Fig. 5. The synthesized hard negative samples were more similar to the positive samples than the randomly chosen negative samples, indicating that our proposed negative sampling approach was effective.

**Table 4**
The ablation studies, in which the negative sampling (w/o Neg), the residual networks (w/o Res) or the final pooling (w/o Pool) were removed, where w/o was an abbreviation for without, and the impact of KGRNS on the interpolation operation of negative sampling using uniform sampling (KGRNS-uniform) or gate mechanism (KGRNS-gate), where KGRNS-gate was the model we proposed in this paper.

| | Amazon-book | | Last-FM | | Alibaba-iFashion | |
|---|---|---|---|---|---|---|
| | *recall* | *ndcg* | *recall* | *ndcg* | *recall* | *ndcg* |
| w/o Neg | 0.1671 | 0.0910 | 0.0947 | 0.0836 | 0.1123 | 0.0698 |
| w/o Res | 0.1791 | 0.0987 | 0.1134 | 0.0998 | 0.1267 | 0.0814 |
| w/o Pool | 0.1802 | 0.1000 | 0.1096 | 0.0985 | 0.1274 | 0.0816 |
| KGRNS-uniform | 0.1802 | 0.1001 | 0.1085 | 0.0977 | 0.1272 | 0.0815 |
| KGRNS-gate | 0.1820 | 0.1006 | 0.1117 | 0.1017 | 0.1302 | 0.0832 |

**Table 5**
User embedding with or without residual connections on Alibaba-iFashion dataset with different $K$ values on four evaluation metrics. The bolded indicate the optimal results with or without residual connections at different $K$ values.

| User embedding | $K$ value | Alibaba-iFashion | | | |
|---|---|---|---|---|---|
| | | *recall@k* | *ndcg@k* | *precision@k* | *hit rate@k* |
| With residual | $K = 10$ | 0.0834 | **0.0649** | 0.0279 | 0.2438 |
| | $K = 20$ | 0.1247 | 0.0798 | 0.0210 | 0.3414 |
| | $K = 30$ | 0.1550 | 0.0891 | 0.0175 | 0.4056 |
| | $K = 50$ | 0.1999 | 0.1014 | 0.0136 | 0.4924 |
| | $K = 100$ | 0.2737 | 0.1188 | 0.0094 | 0.6096 |
| Without residual | $K = 10$ | **0.0875** | 0.0600 | **0.0294** | **0.2545** |
| | $K = 20$ | **0.1302** | **0.0832** | **0.0220** | **0.3544** |
| | $K = 30$ | **0.1613** | **0.0930** | **0.0182** | **0.4186** |
| | $K = 50$ | **0.2074** | **0.1055** | **0.0141** | **0.5047** |
| | $K = 100$ | **0.2820** | **0.1232** | **0.0097** | **0.6211** |

The *recall* on the Last-FM dataset w/o Res was higher than that of KGRNS, which was attributed to the differences of this dataset. The KG of the Last-FM was mainly converted from albums, songs, and artists, and the design of the residual operation instead reduced its performance. The other two datasets, however, benefitted from the design of residual connections. Notably, our KGRNS model only used residual connections in item embedding, not in user embedding. Taking the Alibaba-iFashion dataset as an example, adding residual connections to the user embedding process leads to performance degradation, as shown in Table 5. The Alibaba-iFashion dataset is dominated by first-order connections. The quality of users' first-order neighbors is stumpy, and the item that users have interacted before does not mean they will interact again in the future. Adding residual connections in user embedding will affect the final embedding representation. In addition, using residual connections in user embedding will reduce the relevance of closely related users, which affects recommendation performance.

Table 4 shown that KGRNS model with sum pooling (KGRNS-gate) outperformed the KGRNS model without sum pooling (w/o Pool). It illustrated that the sum pooling operation is effective for our KGRNS model and demonstrated again that the residual connection was enough to combine necessary information from the previous layers. Although residual connections can transfer the previous embedding representation to the next layer, the output of different GNN layers emphasizes the order of connection information differently. By utilizing pooling, we not only enrich the embeddings used for prediction, but also allow controlling propagation strength by adjusting $L$.

We also designed an ablation experiment KGRNS-uniform to demonstrate the effectiveness of the gate operation for the interpolation operation. Table 4 summarizes the experimental findings. The gate operation clearly yielded superior recommendation performances on all three datasets when compared to the uniform sampling method, indicating that the gate operation was

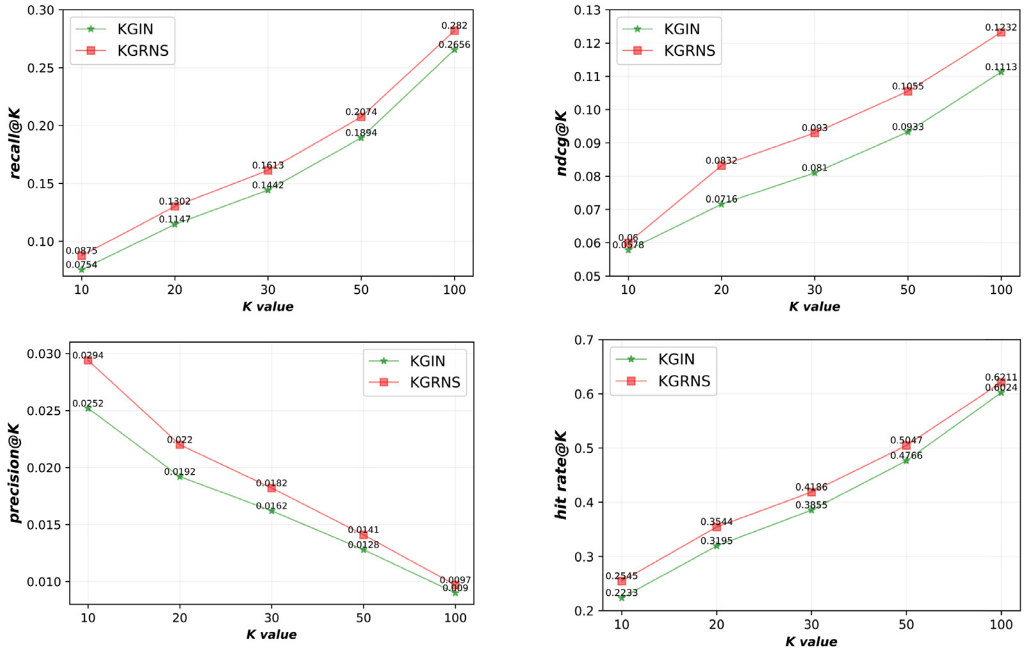
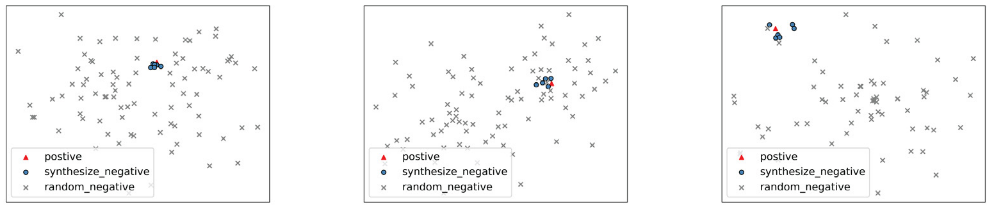


**Fig. 4.** KGRNS model and KGIN model performance comparison of four evaluation metrics (*recall@K*, *ndcg@K*, *precision@K* and *hit rate@K*) with five *K* values (10, 20, 30, 50, 100) on Alibaba-iFashion dataset.



(a) Amazon-Book (b) Last-FM (c) Alibaba-iFashion

**Fig. 5.** Visual representation of positive samples (positive), random negative samples (random_negative), and synthesized hard negative samples (synthesize_negative) on (a) Amazon-Book, (b) Last-FM, and (c) Alibaba-iFashion datasets.

essential for providing higher quality hard negative samples for recommendation system.

The gate operation boosted the performance of KGRNS on the last-FM and Alibaba-iFashion datasets more than on the Amazon-Book dataset. We ascribe the performance gap to complicated high-order connections and numerous relationships. The uniform sampling method generated a sufficient quantity and quality of hard negative samples to train the dataset; therefore, the gate operation had little impact.

### *5.3.2. Effect of number of GNN layers L*

We examined the effect of the number of GNN aggregation layers on the effectiveness of the model. The information conveyed by long-range connections can be included in the node representation by stacking more layers. Here, we modified $L$ in the scope of $\{1, 2, 3\}$, and the results are summarized in Fig. 6. We observed the following:

- In most cases, increasing the model depth improved the prediction results. In particular, as shown in Figs. 6(a) and 6(b), KGRNS-3 achieved significant improvements over KGRNS-1 on all datasets. We attribute this improvement to two factors:
  (1) By stacking additional layers and exploring more correlated entities connected by KG, we may gain a better view of user interests. KGRNS-1 only utilizes the one-hop connectivity, while KGRNS-3 reveals third-order connections.
  (2) Longer relational paths allowed more user-related information to be obtained to analyze users' preferences for items better.
- Fig. 6 shows that the results of KGRNS-2 were worse than those of KGRNS-1 for Alibaba-iFashion. This was due to most of the KG triples of Alibaba-iFashion had first-order connectivity of items, which was captured by KGRNS-1. Moreover, KGRNS-3 exhibited a worse performance than KGRNS-2 on Amazon-book. It again demonstrated that the GNN could lead to over-smoothing problems, especially on datasets with dense U–I interactions.
- KGRNS-3 performed better than other baselines on Alibaba-iFashion. It proved that the residual network was effective in alleviating the over-smoothing problem of GNNs, particularly on datasets with more first-order connections. Therefore, KGRNS-3 could recognize the boundaries of positive and negative samples to generate appropriate node representations.
- KGRNS-3 worked stronger on the Last-FM dataset than on Amazon-Book, although Amazon-Book had a larger set of KG relations than Last-FM. It was attributed to the differences between the two datasets. KGs in Amazon-Book contained noisy relationships unrelated to user behavior.

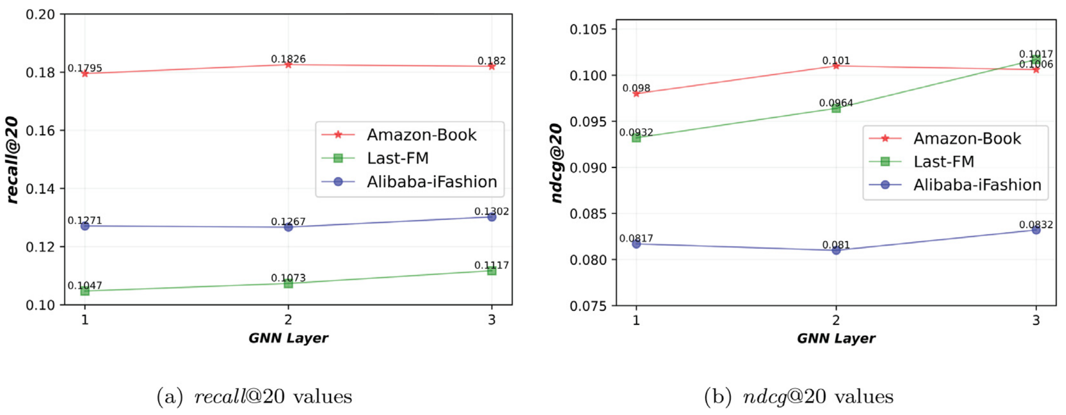


**Fig. 6.** Performance comparison while varying the number of GNN layers from 1 to 3 on the three datasets: Amazon-Book, Last-FM, and Alibaba-iFashion. Best viewed in color. (a) *recall*@20 values of the three datasets with different GNN layers. (b) *ndcg*@20 values of the three datasets with different GNN layers.

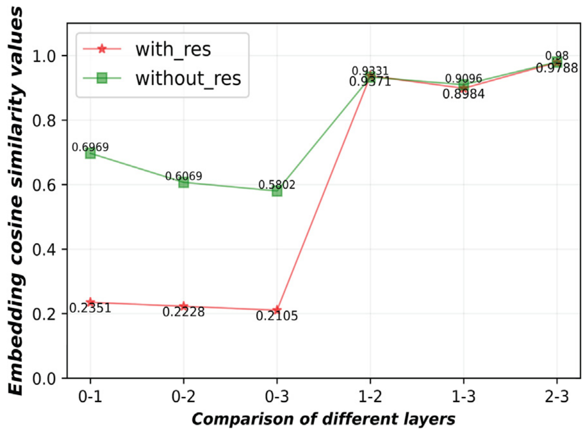


**Fig. 7.** The item embedding cosine similarity between each GNN layers on Amazon-Book. '$a - b$' denotes the similarity of $e_i^{(a)}$ and $e_i^{(b)}$. Specifically, '0-1' denotes the similarity of $e_i^{(0)}$ and $e_i^{(1)}$. A larger similarity value indicates a smaller difference between the two embeddings. The maximum value is 1.

### 5.3.3. *Embedding similarity metric*

We designed an embedding similarity metric that uses cosine similarity to calculate the similarity of item embeddings from different layers. The results on Amazon-Book dataset were shown in Fig. 7.

- The decreasing similarity value of '0-1', '0-2', and '0-3' indicates that the embedding of items becomes increasingly discriminative as the number of GNN layers increases. KGRNS model with residuals has smaller similarity values than the model without residuals, demonstrating that the residual connection we designed can alleviate the over-smoothing problem caused by GNN.
- '1-3' is smaller than '1-2' of KGRNS with residuals also demonstrate that residual connection can further alleviate the over-smoothing problem. Notably, the values of '1-2' and '1-3' are overall larger than that of '0-1', '0-2', and '0-3', demonstrating that there indeed exists an over-smoothing problem as the number of GNN layers increases.
- As for '2-3', regardless of the KGRNS model with or without residual connections, the similarity between the second and third layers is close to 1, which means that using three layers of GNN is sufficient, and increasing the number of layers will not have a great performance improvement.

## 6. Conclusions and future work

In this paper, we designed a model named KGRNS. We designed a new negative sampling method that allows the model to sample high-quality negative samples that are both responsive to users' preferences and informative, improving the ability of KGRNS to distinguish the boundaries between positive and negative samples. We also designed residual connections on the output of each GNN layer and performed sum pooling operation to alleviate the over-smoothing issue of deep GNN models. The KGRNS performed better than state-of-the-art models on all three datasets.

A KG-based recommendation is often treated as a supervised task. There is little supervision to ensure that a high-quality representation is provided. In the follow-up research work, we will reveal internal relationships between data samples through self-supervised tasks by probing self-supervised learning in RS. Furthermore, we would like to explore and expand on the methods designed in KGRNS for pre-training graphs to develop robust generic representations. Finally, we will focus on hypergraph-based recommendation systems in our future work.

## CRediT authorship contribution statement

**Yuanyuan Liu:** Conceptualization, Methodology, Software, Writing – original draft, Investigation. **Zhaoqian Zhong:** Writing – review & editing. **Chao Che:** Validation, Supervision, Funding acquisition. **Yongjun Zhu:** Data curation, Visualization.

## 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 was supported by the National Natural Science Foundation of China (No. 62076045, 62172074, 61772110), the High-Level Talent Innovation Support Program (Young Science and Technology Star) of Dalian (No. 2021RQ066), Subject of Liaoning Provincial Department of Education (LZ2019002), Program of Introducing Talents of Discipline to Universities (Plan 111), China (No. B20070) and Key Program of Liaoning Traditional Chinese Medicine Administration(LNZYXZK201910).

## References

[1] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.-S. Chua, Neural collaborative filtering, in: Proceedings of the 26th International Conference on World Wide Web, 2017, pp. 173–182.

[2] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (8) (2009) 30–37.

[3] M. Weimer, A. Karatzoglou, A. Smola, Adaptive collaborative filtering, in: Proceedings of the 2008 ACM Conference on Recommender Systems, 2008, pp. 275–282.

[4] H. Zhang, F. Shen, W. Liu, X. He, H. Luan, T.-S. Chua, Discrete collaborative filtering, in: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2016, pp. 325–334.

[5] D. Liang, R.G. Krishnan, M.D. Hoffman, T. Jebara, Variational autoencoders for collaborative filtering, in: Proceedings of the 2018 World Wide Web Conference, 2018, pp. 689–698.

[6] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of the 10th International Conference on World Wide Web, 2001, pp. 285–295.

[7] Z.-D. Zhao, M.-S. Shang, User-based collaborative-filtering recommendation algorithms on hadoop, in: 2010 Third International Conference on Knowledge Discovery and Data Mining, IEEE, 2010, pp. 478–481.

[8] H. Wang, F. Zhang, J. Wang, M. Zhao, W. Li, X. Xie, 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.

[9] X. Wang, X. He, Y. Cao, M. Liu, T.-S. Chua, Kgat: Knowledge graph attention network for recommendation, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 950–958.

[10] 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.

[11] Z. Wang, G. Lin, H. Tan, Q. Chen, X. Liu, CKAN: collaborative knowledge-aware attentive network for recommender systems, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 219–228.

[12] X. Wang, T. Huang, D. Wang, Y. Yuan, Z. Liu, X. He, T.-S. Chua, Learning intents behind interactions with knowledge graph for recommendation, in: Proceedings of the Web Conference 2021, 2021, pp. 878–887.

[13] L. Cui, D. Lee, KETCH: Knowledge graph enhanced thread recommendation in healthcare forums, in: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 492–501, http://dx.doi.org/10.1145/3477495.3532008.

[14] S.-J. Park, D.-K. Chae, H.-K. Bae, S. Park, S.-W. Kim, Reinforcement learning over sentiment-augmented knowledge graphs towards accurate and explainable recommendation, in: Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, WSDM '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 784–793, http://dx.doi.org/10.1145/3488560.3498515.

[15] W. Hamilton, P. Bajaj, M. Zitnik, D. Jurafsky, J. Leskovec, Embedding logical queries on knowledge graphs, Adv. Neural Inf. Process. Syst. 31 (2018).

[16] B. Hu, C. Shi, W.X. Zhao, P.S. Yu, Leveraging meta-path based context for top-n recommendation with a neural co-attention model, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 1531–1540.

[17] X. Wang, D. Wang, C. Xu, X. He, Y. Cao, T.-S. Chua, Explainable reasoning over knowledge graphs for recommendation, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 5329–5336.

[18] Y. Xian, Z. Fu, S. Muthukrishnan, G. De Melo, Y. Zhang, Reinforcement knowledge graph reasoning for explainable recommendation, in: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2019, pp. 285–294.

[19] H. Wang, F. Zhang, M. Zhang, J. Leskovec, M. Zhao, W. Li, Z. Wang, Knowledge-aware graph neural networks with label smoothness regularization for recommender systems, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 968–977.

[20] H. Wang, M. Zhao, X. Xie, W. Li, M. Guo, Knowledge graph convolutional networks for recommender systems, in: The World Wide Web Conference, WWW '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 3307–3313, http://dx.doi.org/10.1145/3308558.3313417.

[21] K. Zhao, Y. Zheng, T. Zhuang, X. Li, X. Zeng, Joint learning of E-commerce search and recommendation with a unified graph neural network, in: Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, WSDM '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 1461–1469, http://dx.doi.org/10.1145/3488560.3498414.

[22] W. Fan, X. Liu, W. Jin, X. Zhao, J. Tang, Q. Li, Graph trend filtering networks for recommendation, in: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 112–121, http://dx.doi.org/10.1145/3477495.3531985.

[23] R. Yu, D. Ye, Z. Wang, B. Zhang, A.M. Oguti, J. Li, B. Jin, F. Kurdahi, CFFNN: Cross feature fusion neural network for collaborative filtering, IEEE Trans. Know. Data Eng. 34 (10) (2022) 4650–4662, http://dx.doi.org/10.1109/TKDE.2020.3048788.

[24] F. Ojo, R.A. Rossi, J. Hoffswell, S. Guo, F. Du, S. Kim, C. Xiao, E. Koh, VisGNN: Personalized visualization recommendationvia graph neural networks, in: Proceedings of the ACM Web Conference 2022, WWW '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 2810–2818, http://dx.doi.org/10.1145/3485447.3512001.

[25] V.P. Dwivedi, C.K. Joshi, T. Laurent, Y. Bengio, X. Bresson, Benchmarking graph neural networks, 2020, arXiv preprint arXiv:2003.00982.

[26] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, Adv. Neural Inf. Process. Syst. 30 (2017).

[27] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, 2016, arXiv preprint arXiv:1609.02907.

[28] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, 2017, arXiv preprint arXiv:1710.10903.

[29] W. Guo, Y. Yang, Y. Hu, C. Wang, H. Guo, Y. Zhang, R. Tang, W. Zhang, X. He, Deep graph convolutional networks with hybrid normalization for accurate and diverse recommendation, in: Proceedings of 3rd Workshop on Deep Learning Practice for High-Dimensional Sparse Data with KDD, 2021.

[30] J. Wang, K. Ding, L. Hong, H. Liu, J. Caverlee, Next-item recommendation with sequential hypergraphs, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 1101–1110, http://dx.doi.org/10.1145/3397271.3401133.

[31] R. Togashi, M. Otani, S. Satoh, Alleviating cold-start problems in recommendation through pseudo-labelling over knowledge graph, in: Proceedings of the 14th ACM International Conference on Web Search and Data Mining, 2021, pp. 931–939.

[32] X. Hao, Y. Liu, R. Xie, K. Ge, L. Tang, X. Zhang, L. Lin, Adversarial feature translation for multi-domain recommendation, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 2964–2973.

[33] X. Wang, Y. Xu, X. He, Y. Cao, M. Wang, T.-S. Chua, Reinforced negative sampling over knowledge graph for recommendation, in: Proceedings of the Web Conference 2020, 2020, pp. 99–109.

[34] T. Huang, Y. Dong, M. Ding, Z. Yang, W. Feng, X. Wang, J. Tang, MixGCF: An improved training method for graph neural network-based recommender systems, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 665–674.

[35] Y. Chen, X. Wang, M. Fan, J. Huang, S. Yang, W. Zhu, Curriculum meta-learning for next POI recommendation, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 2692–2702.

[36] R. Ying, R. He, K. Chen, P. Eksombatchai, W.L. Hamilton, J. Leskovec, Graph convolutional neural networks for web-scale recommender systems, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 974–983.

[37] J. Ding, Y. Quan, X. He, Y. Li, D. Jin, Reinforced negative sampling for recommendation with exposure data, in: IJCAI, 2019, pp. 2230–2236.

[38] X. Wang, X. He, M. Wang, F. Feng, T.-S. Chua, Neural graph collaborative filtering, in: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2019, pp. 165–174.

[39] I. Spinelli, S. Scardapane, A. Uncini, Adaptive propagation graph convolutional network, IEEE Trans. Neural Netw. Learn. Syst. 32 (10) (2020) 4755–4760.

[40] T.N. Kipf, M. Welling, Variational graph auto-encoders, 2016, arXiv preprint arXiv:1611.07308.

[41] S. Wu, Y. Zhang, C. Gao, K. Bian, B. Cui, Garg: anonymous recommendation of point-of-interest in mobile networks by graph convolution network, Data Sci. Eng. 5 (4) (2020) 433–447.

[42] G. Cui, J. Zhou, C. Yang, Z. Liu, Adaptive graph encoder for attributed graph embedding, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 976–985.

[43] X. Zhang, H. Liu, Q. Li, X.-M. Wu, Attributed graph clustering via adaptive graph convolution, 2019, arXiv preprint arXiv:1906.01210.

[44] X. Chen, K. Xiong, Y. Zhang, L. Xia, D. Yin, J.X. Huang, Neural feature-aware recommendation with signed hypergraph convolutional network, ACM Trans. Inf. Syst. 39 (1) (2020) http://dx.doi.org/10.1145/3423322.

[45] H. Wu, M.K. Ng, Hypergraph convolution on nodes-hyperedges network for semi-supervised node classification, ACM Trans. Knowl. Discov. Data 16 (4) (2022) http://dx.doi.org/10.1145/3494567.

[46] H. Wu, Y. Yan, M.K. Ng, Hypergraph collaborative network on vertices and hyperedges, IEEE Trans. Pattern Anal. Mach. Intell. (2022) 1, http://dx.doi.org/10.1109/TPAMI.2022.3178156.
[47] Y. Gao, Z. Zhang, H. Lin, X. Zhao, S. Du, C. Zou, Hypergraph learning: Methods and practices, IEEE Trans. Pattern Anal. Mach. Intell. 44 (5) (2022) 2548–2566, http://dx.doi.org/10.1109/TPAMI.2020.3039374.
[48] Y. Zhao, X. Wang, J. Chen, Y. Wang, W. Tang, X. He, H. Xie, Time-aware path reasoning on knowledge graph for recommendation, ACM Trans. Inf. Syst. (2022) http://dx.doi.org/10.1145/3531267, Just Accepted.
[49] Y. Zhang, X. Wu, Q. Fang, S. Qian, C. Xu, Knowledge-enhanced attributed multi-task learning for medicine recommendation, ACM Trans. Inf. Syst. (2022) http://dx.doi.org/10.1145/3527662, Just Accepted.
[50] Y. Feng, H. You, Z. Zhang, R. Ji, Y. Gao, Hypergraph neural networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, 33, (01) 2019, pp. 3558–3565.
[51] J. Jin, J. Qin, Y. Fang, K. Du, W. Zhang, Y. Yu, Z. Zhang, A.J. Smola, An efficient neighborhood-based interaction model for recommendation on heterogeneous graph, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 75–84.
[52] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, BPR: Bayesian personalized ranking from implicit feedback, 2012, arXiv preprint arXiv:1205.2618.
[53] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, M. 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, 2020, pp. 639–648.
[54] L. Chen, L. Wu, R. Hong, K. Zhang, M. Wang, Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 27–34.
[55] J. Qiu, J. Tang, H. Ma, Y. Dong, K. Wang, J. Tang, Deepinf: Social influence prediction with deep learning, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 2110–2119.
[56] C.-K. Hsieh, L. Yang, Y. Cui, T.-Y. Lin, S. Belongie, D. Estrin, Collaborative metric learning, in: Proceedings of the 26th International Conference on World Wide Web, 2017, pp. 193–201.
[57] F. Schroff, D. Kalenichenko, J. Philbin, Facenet: A unified embedding for face recognition and clustering, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 815–823.
[58] K. Sohn, Improved deep metric learning with multi-class n-pair loss objective, Adv. Neural Inf. Process. Syst. 29 (2016).
[59] Q. Zhu, X. Zhou, J. Wu, J. Tan, L. Guo, A knowledge-aware attentional reasoning network for recommendation, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 6999–7006.
[60] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[61] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.
[62] M. Schlichtkrull, T.N. Kipf, P. Bloem, R.v.d. Berg, I. Titov, M. Welling, Modeling relational data with graph convolutional networks, in: European Semantic Web Conference, Springer, 2018, pp. 593–607.
[63] W. Krichene, S. Rendle, On sampled metrics for item recommendation, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 1748–1757.
[64] J.-H. Yang, C.-M. Chen, C.-J. Wang, M.-F. Tsai, HOP-rec: high-order proximity for implicit recommendation, in: Proceedings of the 12th ACM Conference on Recommender Systems, 2018, pp. 140–144.
[65] J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, X. Xie, Self-supervised graph learning for recommendation, in: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021, pp. 726–735.
[66] H. Chen, C.-C.M. Yeh, F. Wang, H. Yang, Graph neural transport networks with non-local attentions for recommender systems, in: Proceedings of the ACM Web Conference 2022, 2022, pp. 1955–1964.