Information Sciences 647 (2023) 119465

Contents lists available at ScienceDirect

Information Sciences

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

To see further: Knowledge graph-aware deep graph convolutional network for recommender systems

Fei Wang [a,b,c], Zhi Zheng [a], Yongjun Zhang [a,*], Yansheng Li [a,d,**], Kun Yang [e], Chenming Zhu [f]

[a] School of Remote Sensing and Information Engineering, Wuhan University, Wuhan, Hubei, China
[b] Changjiang Spatial Information Technology Engineering Co., Ltd, Wuhan, China
[c] Water Resources Information Perception and Big Data Engineering Research Center of Hubei Province, Wuhan, China
[d] Hubei Luojia Laboratory, Wuhan, Hubei, China
[e] Basic Geographic Information Center of Guizhou Province, Guiyang, China
[f] The First Institute of Photogrammetry and Remote Sensing, Ministry of Natural Resources, China

ARTICLE INFO

Keywords:
Recommender systems
Collaborative filtering
Knowledge graph
Deep graph convolutional network

ABSTRACT

Applying a graph convolutional network (GCN) or its variants to user-item interaction graphs is one of the most commonly used approaches for learning the representation of users and items in modern recommender systems. However, these models perform poorly in recommending high-order items due to the over-smoothing when the GCN deepens and the lack of connectivity between the cold-start items and the target user. To improve the ability to recommend higher-order items, we propose our Knowledge graph-aware Deep Graph Convolutional Network with Initial Residual Connection (KDGCN-IC) and Knowledge graph-aware Deep Graph Convolutional Network with Dense Connection (KDGCN-DC) methods. First, we introduce an item knowledge graph to rebuild the connectivity between nodes in the user-item graph, especially the users and the cold-start items. Then, we present our methods for improving the information flow by reusing features to alleviate the over-smoothing issue to deepen the GCN to capture the higher-order collaborative signal. More specifically, KDGCN-IC reuses the initial feature information, and KDGCN-DC reuses the feature information of each layer. Extensive experiments demonstrated that our models were able to achieve substantial improvement over state-of-the-art models in so far as their recommendation accuracy and higher-order recommendations.

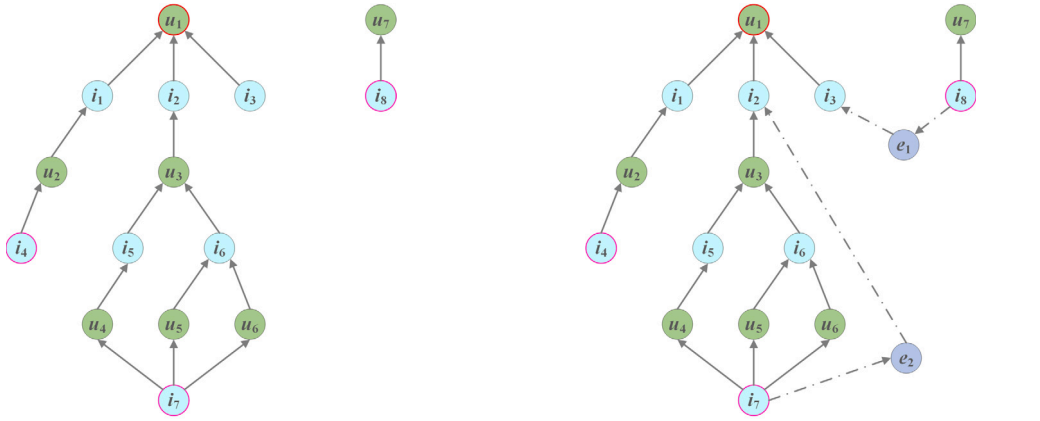
1. Introduction

Recommender systems attempt to alleviate information overload by helping users capture the items of interest among numerous candidates while capturing the user's preferences and recommending items with which they may interact based on the user's interests and the characteristics of items [42]. In general, the user's interests and the item's characteristics are represented by embedding vectors. Thus, a powerful recommender system must be capable of learning appropriate user/item vectors and achieving

* Corresponding author.
** Corresponding author at: School of Remote Sensing and Information Engineering, Wuhan University, Wuhan, Hubei, China.
E-mail addresses: flyking@whu.edu.cn (F. Wang), zhengzhi@whu.edu.cn (Z. Zheng), zhangyj@whu.edu.cn (Y. Zhang), yansheng.li@whu.edu.cn (Y. Li), 413135739@qq.com (K. Yang), 295469081@qq.com (C. Zhu).

https://doi.org/10.1016/j.ins.2023.119465
Received 4 May 2022; Received in revised form 11 May 2023; Accepted 5 August 2023

Available online 10 August 2023



(a) user-item interaction graph (b) user-item interaction graph with item knowledge graph

Fig. 1. An illustration of (a) the user-item interaction graph and (b) the user-item interaction graph with the item knowledge graph. A solid line indicates that the user interacted with the item. A dotted line indicates that there is a relation between the entities. i_7 is a high-order item of u_1, and i_8 is a cold-start item of u_1.

recommendations by calculating the similarities between them. An efficient technique for this purpose is collaborative filtering (CF), which learns the representation of users and items by reconstructing their historical interactions based on the assumption that users with similar behaviors have similar preferences for items [10]. Various CF-based methods [20,29,14] have been proposed and have achieved tremendous success. However, their applications are limited because they implicitly encode user-item interactions and ignore higher-order structural information [7].

Recently, advances have been made in graph convolutional networks (GCNs) [18,12,32], and are providing a solution to the above limitations in recommender systems. GCNs are designed to aggregate neighbor embeddings based on signal processing theory. By stacking multiple aggregation layers, higher-order neighbor features can be propagated to lower-order nodes iteratively [45]. By applying GCNs to user-item interactions that can be represented by bipartite graphs, their interactions can be encoded explicitly. GC-MC [1] regards the recommendation as a link prediction problem and designs a graph-based auto-encoder to learn the representation of users and items. PinSage [47] combines random walk and L_1 normalization to sample the top-k important neighbors and utilizes graph convolutions to aggregate their embeddings to generate node embeddings. NGCF [38] refines the representation of users and items following the standard design of GCN: neighborhood aggregation, feature transformation, and nonlinear activation. LightGCN [13] determines that feature transformation and nonlinear activation are redundant for CF and simplifies NGCF by removing them. LightGCN can achieve state-of-the-art performance while greatly reducing the trainable parameters. Hypergraphs are also a popular choice when capturing high-order connectivity [24], however, the sparsity of hypergraphs leads to modeling difficulties.

Although LightGCN has greatly improved performance, experiments on LightGCN demonstrate that it is difficult for LightGCN to accurately recommend the items to the target users that are high-order away from them. The same situation also occurs in other GCN-based models. We attribute this to the following two reasons: (1) Over-smoothing issue. As the number of GCN layers deepens, the node representation tends to converge to a certain value leading to the model's inability to identify the item of interest. These GCN-based models achieve their best performance at shallow layers, e.g., GC-MC [1] is a one-layer model, PinSage [47] has two layers, and NGCF [38] and LightGCN [13] are less than four layers. However, when the number of layers is shallow, the representation of higher-order items cannot be propagated to the user, so there is a lack of similarity between the user and the interested high-order items, which is the basis of the recommendation. (2) Lack of connectivity. In user-item bipartite graphs, there is sometimes no path to connect items of interest to their target users, which may also result in the representation of these items not being propagated to users. Similar to the general cold-start problem, we consider the lack of connectivity as a specific cold-start problem within the user-item graph, because although these items are in the dataset, they are new to the target users, and we refer to these items as cold-start items.

Example. Fig. 1(a) shows the tree structure of a user-item interaction graph where the user of interest for recommendation is u_1 (labeled with the red circle in the figure). Assume that i_8 is highly similar to i_3 in attributes. We analyze the possibility of recommending i_4, i_7, and i_8 (labeled with the purple circle in the figure). There is one path connecting i_4 and u_1 (i.e., $i_4 \rightarrow u_2 \rightarrow i_1 \rightarrow u_1$), three paths connecting i_7 and u_1 (i.e., $i_7 \rightarrow u_4 \rightarrow i_5 \rightarrow u_3 \rightarrow i_2 \rightarrow u_1$, $i_7 \rightarrow u_5 \rightarrow i_6 \rightarrow u_3 \rightarrow i_2 \rightarrow u_1$ and $i_7 \rightarrow u_6 \rightarrow i_6 \rightarrow u_3 \rightarrow i_2 \rightarrow u_1$) and no path between i_8 and u_1. From the view of high-order connectivity, i_4 is most likely to be recommended to u_1 because i_4 is the closest. From the view of feature aggregation, i_7 is more likely to be recommended because more of its feature is propagated to u_1 in a deep GCN. From the view of attributes, i_8 also should be recommended because it is similar to i_3, which interacted with u_1 before. However, in a shallow layer model, without introducing side information, only i_4 is recommended to u_1.

In this work, we present extensive studies to improve the performance of high-order recommendation. First, we introduce an item knowledge graph to enrich the user-item interaction graph (Fig. 1(b)), in which items are connected by attributes or characteristics (e.g., e_1 and e_2). In this way, similar items can be closer (e.g., i_2 and i_7), and users are likely to see items that are of interest but not connected (e.g., u_1 and i_8). TMER [4] utilizes item-item paths specially designed on the item knowledge graph to model user-item relationships, however, these paths are domain specific. KGAT and KGCN [37,36] also introduce item knowledge graphs; however,

they are shallow models and show poor performance in recommending higher-order items as does LightGCN. Second, we design two deep GCN models, named Knowledge graph-aware Deep Graph Convolutional Network with Initial Residual Connection (KDGCN-IC) and Knowledge graph-aware Deep Graph Convolutional Network with Dense Connection (KDGCN-DC), to aggregate higher-order features, where KDGCN-IC contains the initial residual connection and holistic connection, and KDGCN-DC contains the dense connection and holistic connection. The initial residual connection adds a fraction of the initial representation into each subsequent layer to ensure that each node's final representation retains a part of its initial information. The holistic connection propagates the representation at each layer to the last layer to generate the integrated representation, ensuring that the final representation contains all the features captured by each layer. The purpose of the dense connection is to determine a fraction of the representation at each of its preceding layers flows into the layer to improve the information flow and facilitate the efficient reuse of features.

The contributions of our work are as follows:

- We experimentally prove that the current GCN-based models show obvious limitations in recommending higher-order items to users, and we empirically point out that the main reasons are the over-smoothing issue and the lack of connectivity issue. To the best of our knowledge, we are the first to reveal and attempt to address the limitation.
- We propose two knowledge graph-aware deep GCNs, which incorporate feature reuse by employing initial residual connection, dense connection, and holistic connection, and connectivity reconstruction by introducing item knowledge graph for alleviating the over-smoothing issue and the lack of connectivity issue, respectively.
- Extensive experiments on three public datasets demonstrate the superiority of our models over state-of-the-art models in terms of conventional and higher-order recommendations.

The remainder of this paper is organized as follows. Section 2 introduces the most related literature. Section 3 introduces our task. Section 4 describes our methods in detail. Section 5 presents our experimental result and analysis. Section 6 discusses our conclusions and future work.

2. Related work

In this section, we review the existing literature pertaining to graph convolutional network-based recommender systems and deep graph convolutional networks, which are most relevant to and greatly inspired our work.

2.1. Graph convolutional networks for recommender systems

The main task of current recommender systems is to learn the representation of users and items. GCN [18] has greatly contributed to the development of recommender systems due to its powerful capabilities in graph structure modeling and node embedding learning. GC-MC [1] learns the node embedding by applying a one-layer GCN to the user-item bipartite graph. However, GC-MC can only learn the direct connections between nodes and cannot exploit higher-order connectivity. PinSage [47] simulates effective random walks to generate the local graph neighborhood for each node and iteratively aggregates feature information from the high-order neighbors by stacking multiple layers. SR-GNN [43] uses GCN to capture complex information transitions between items on the construction graph. However, PinSage and SR-GNN aggregate the features in the item-item graph but ignores the user-item connectivity. NGCF [38] captures the high-order connectivity between users and items by performing a standard multi-layer GCN on the user-item graph. However, feature transformation and nonlinear activation contained in NGCF are demonstrated to be redundant for recommendation and negatively affect performance. LR-GCCF [5] removes the nonlinear activation, and LightGCN [13] and SGL-ED [41] further remove both components. The above GCN-based models cannot be extended to deep structures since they suffer from the over-smoothing issue. DGCN-HN [11] utilizes a residual connection during feature aggregation to deepen GCNs, and UltraGCN [28] even skips the message passing but just directly approximates the limit of infinite layer graph convolutions through a constraint loss, but they cannot address the lack of connectivity issue. In addition, DGCN-HN retains many additional trainable parameters.

Using GCNs to explore information on the item knowledge graph for recommendations is another important direction, as the knowledge graph can provide rich profile features for items. KGCN [36] and KGNN-LS [35] apply GCN to item knowledge graphs directly to mine the similarities between items as well as the user's preferences to relations, but they ignore the connectivity between the users and the items. KGAT [37] merges the user-item interaction graph and item knowledge graph into an integrated graph and learns the representation of nodes by alternately training the recommendation task and the knowledge graph embedding task with GCN and TransR [25] respectively. In this way, the user features can be enriched by the item knowledge graph. CKAN [39] utilizes a knowledge-aware attention mechanism to distinguish the contributions of various neighbors in the feature aggregation process. CGAT [27] aggregates information of first-order neighbors and higher-order neighbors of entities in the item knowledge graph by using the graph attention mechanism and biased random walk-based GRU, respectively. Although KGAT, CKAN, and CGAT achieve noteworthy performance, they introduce additional weight matrices. In addition, these mentioned knowledge graph-aware methods mitigate the lack of connectivity issue by introducing item knowledge graphs to reconstruct paths between users and projects, however, they suffer from over-smoothing as more layers are stacked.

2.2. Deep graph convolutional networks

GCN and its variants have achieved enormous success in various graph-based applications, such as node classification [18], graph classification [22], link prediction [17], oil spill detection [26], and recommender systems [47]. However, these models achieve

Table 1
Distribution of the top 20 recommended items to the sample users *w.r.t.* orders with the target user on Book-Crossing. The first row refers to the order between the target user and the recommended item, and user$_i$ is the i_{th} sampled user.

user \ order	3	5	7	-1
user$_1$	19	1	0	0
user$_2$	20	0	0	0
user$_3$	20	0	0	0
user$_4$	20	0	0	0
user$_5$	20	0	0	0

their best performance at the shallow layers and their performance degrades as the layers deepen because of over-smoothing [23]. Many attempts have been made to deepen GCN by tackling over-smoothing.

The vanilla GCN [18] applies the residual connections between hidden layers to train a deeper model, but it performs worse when there are three or more layers. PPNP and APPNP [19] relieve over-smoothing by replacing the weight matrix in the vanilla GCN with the Personalized PageRank matrix, but they fail to train a deep model. JK-Net [46] passes the embedding at each of the previous layers to the final layer with different aggregators to prevent over-smoothing and achieves the best performance when the layer is six on some datasets. Although JK-Net is not deep enough, it shows that shallow features are beneficial for alleviating the over-smoothing. DenseNet [21] adds dense connections to the vanilla GCN and successfully builds a 56-layer GCN without losing accuracy. GCNII [6] extends the vanilla GCN with initial residual and identity mapping to prevent over-smoothing, and its performance improves as the model deepens.

In addition to adjusting the network architecture, applying regularization techniques is another solution. DropEdge [31] randomly drops a proportion of the edges from the input graph to alleviate overfitting and over-smoothing. PairNorm [49] makes the nodes distinguishable by ensuring that the total distance of the pairwise features remains constant. NodeNorm [50] is a node normalization technique to scale each node using the statistical characteristics of the node itself for reducing over-smoothing. The above methods achieve their best performance at deep layers (e.g., 32 or 64). MADReg [3] utilizes the gap of mean average distance (MADGap) to estimate the over-smoothness of graph representation and add a MADGap-based regularizer to the objective function to alleviate the over-smoothing issue.

Although the above methods have been applied for tasks like node classification and point cloud semantic segmentation, their performance in recommendation thus is uncertain. GCNII [6] and DenseNet [21] are two models that are structurally similar to our proposed models, but we remove the feature transformation and nonlinear activation during feature aggregation to make the model more suitable for the recommendation task.

3. Preliminaries

Before presenting our proposed methods, we perform studies on LightGCN to explore its ability to recommend higher-order items and then define the related concepts, notations, and our recommendation task.

3.1. Empirical explorations on LightGCN

LightGCN is a light and powerful GCN-based recommendation model. We conducted a study on LightGCN on the Book-Crossing dataset to explore its ability to recommend items of different orders to users.

Order. The order between the user and the item is defined as the minimum number of edges from the user to reach the item. -1 indicates that the user is not connected to the item (i.e., the cold-start item of the user) and is considered as the highest order. For example, in Fig. 1(a), the order of $< u_1, i_6 >$ is 3, and that of $< u_1, i_8 >$ is -1.

We used the LightGCN codes released by NeuRec,[1] and split the dataset into a training set, a validation set, and a test set. We tuned hyperparameters on the validation set. To achieve our goal, we randomly sampled five users and counted the order distribution between each user and twenty items most likely to engage the user. Note that the items observed in the training set that interacted with the user (user's first-order neighbors) were not recommended, and the users' even-order neighborhoods are users. Table 1 shows the results. As can be seen that almost all items recommended to a user belong to the user's third-order neighborhood, i.e., the closest neighbors that can be recommended to the target user, which proves that the order of the item and the user largely determines whether the item is recommended to the user.

Then we split the test set by order to further evaluate LightGCN's ability to recommend higher-order items, and we explored the effect of the number of layers on this ability. Only the number of model layers was changed, while other hyperparameters (such as learning rate, batch size, and regularization coefficient) were fixed. Table 2 reports the results, wherein ord_i indicates the sub-test set

[1] https://github.com/wubinzzu/NeuRec.

Table 2
Performance *w.r.t.* different LightGCN layers on the sub-test sets of Book-Crossing with different user-item orders.

	ord_3		ord_5		ord_-1	
layers	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
1	0.2391	0.1344	0.0056	0.0019	0.0006	0.0002
2	0.2421	0.1374	0.0109	0.0039	0.0011	0.0003
3	0.2484	0.1408	0.0105	0.0037	0.0006	0.0004
4	0.2492	0.1409	0.0124	0.0053	0.0006	0.0002
8	0.2261	0.1340	0.0171	0.0074	0.0000	0.0000
16	0.1752	0.1202	0.0322	0.0171	0.0006	0.0006

containing all the user-item pairs that the item is $i-$order away from the user. Jointly analyzing Table 1 and Table 2, we concluded the following:

- The shallow LightGCN performed poorly in recommending higher-order items, and the deep LightGCN improved the accuracy of recommending higher-order items at the expense of recommending lower-order items. As the number of layers increased, LightGCN's performance increased and then decreased on ord_3 but gradually increased on ord_5.
- LightGCN was unable to recommend the cold-start item to users. As the number of layers increased, LightGCN maintained a poor performance on ord_-1.

From the above observations, we can draw the conclusion that: (1) The limited aggregation layer can only propagate the features of lower-order neighborhoods to the target users, while the features of higher-order items cannot be propagated to them, resulting in the potential recommended higher-order items not being accurately recommended due to the lack of explicit similarity, i.e., the features learned by the shallow LightGCN have neighboring similarity but no holistic similarity. However, suffering from the over-smoothing issue, LightGCN fails to maintain its performance at a deep layer; (2) Even in the deeper models, the representations of cold-start items cannot be propagated to the target users due to the lack of connectivity, resulting in those cold-start items that should be of interest cannot be effectively recommended due to the lack of representational similarity. Therefore, to allow users to obtain the representations of higher-order or cold-start items, it's necessary to introduce side information to reconstruct the user-item connectivity and train a deep GCN-based model.

3.2. Problem formulation

User-Item Bipartite Graph. For recommender purposes, we generally have a group of M users $\boldsymbol{U} = \{u_1, u_2, \dots, u_M\}$, a group of N items $\boldsymbol{I} = \{i_1, i_2, \dots, i_N\}$ and a set of user-item historical interaction records (e.g., rates, purchases, clicks, and views). The interactions can be represented by an undirected bipartite graph $\boldsymbol{G}_1$, which is defined as $\{(u, y_{ui}, i) | u \in \boldsymbol{U}, i \in \boldsymbol{I}\}$, where $y_{ui} = 1$ indicates that user u implicitly engaged with item i before, otherwise $y_{ui} = 0$.

Item Knowledge Graph. The item knowledge graph consists of items and their side information (e.g., attributes or characteristics) is also available in our tasks. This supplementary knowledge enriches the description of an item through its entities and their relationships. For example, a book can be described in detail by its author, category, and publisher. The side information is organized as a knowledge graph $\boldsymbol{G}_2$ in entity-relation-entity triples. $\boldsymbol{G}_2$ is represented by a directed graph, defined as $\{(h, r, t) | h \in \mathcal{E}, r \in \mathcal{R}, t \in \mathcal{E}\}$, where $\mathcal{E} \in \mathbb{R}^O$ is a group of entities and $\mathcal{R} \in \mathbb{R}^Q$ is a group of relations, O and Q are their cardinalities, respectively. Each triple (h, r, t) is real-world knowledge, which means that there is a relation r from head entity h to tail entity t [2]. For example, the triple (*Hemingway, author, For Whom the Bell Tolls*) states the fact that Hemingway wrote the book For Whom the Bell Tolls. Each item in $\boldsymbol{G}_1$ is aligned with an entity in $\boldsymbol{G}_2$, so $\mathcal{E}$ is composed of items $\boldsymbol{I}$ and the non-items $\mathcal{E} \setminus \boldsymbol{I}$ (e.g., item attributes).

Collaborative Heterogeneous Graph. We merge $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ as an integrated graph on which we propagate information. We first represent each observed user behavior in $\boldsymbol{G}_1$ as a triple (user, *Interacts*, items) and update them into $\boldsymbol{G}_2$. Then we simplify each triple in the merged knowledge graph as $(h, hasRelation, t)$, i.e., the relations are unified into a special relation *hasRelation*, because we only consider the connectivity between entities but no longer the relation between them. The integrated graph is defined as an undirected Collaborative Heterogeneous Graph $\boldsymbol{G} = \{(h, y_{ht}, t) | h \in \mathcal{E}', t \in \mathcal{E}'\}$, where $\mathcal{E}' = \mathcal{E} \cap \boldsymbol{U}$, $y_{ht} = 1$ indicates that h connects t.

Task Formulation. The task to be solved can be formulated as:

- **Input**: Collaborative heterogeneous graph $\boldsymbol{G}$ merged and simplified by the user-item bipartite graph $\boldsymbol{G}_1$ and the item knowledge graph $\boldsymbol{G}_2$.
- **Output**: Prediction function $\mathcal{F}(u, i | \boldsymbol{G}, \theta)$ that predicts the probability $\hat{y}_{ui}$ that user u has a preference for item i, where θ is the parameters.

4. Methodology

In this section, we introduce our proposed Knowledge graph-aware Deep Graph Convolutional Network with Initial Residual Connection (KDGCN-IC) and Knowledge graph-aware Deep Graph Convolutional Network with Dense Connection (KDGCN-DC). The

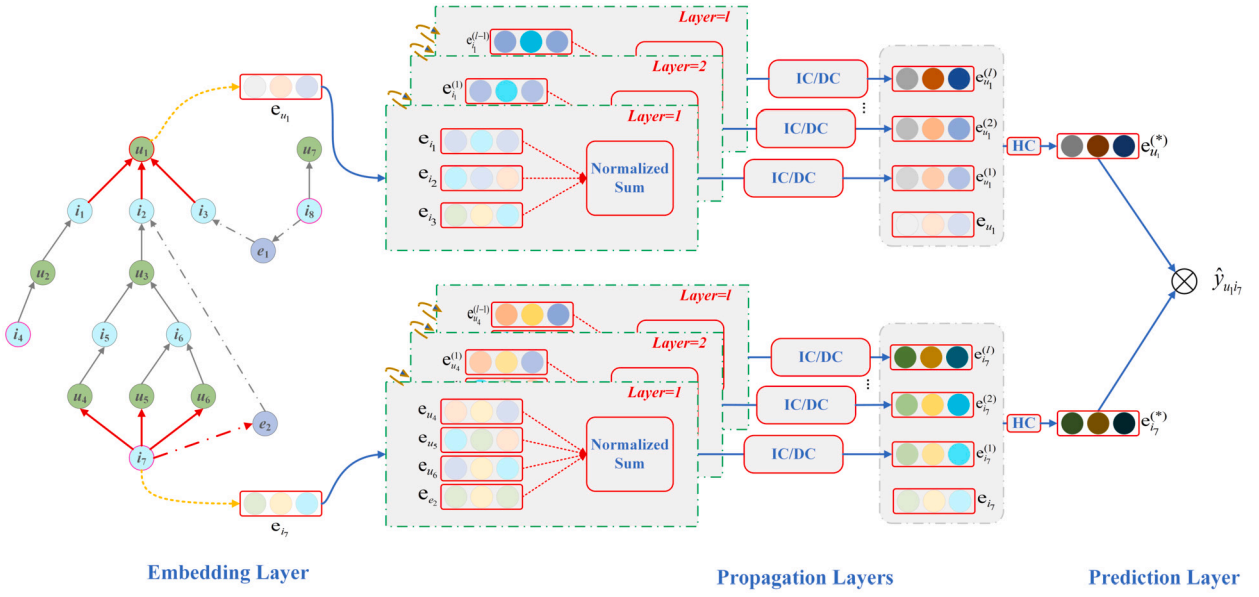


Fig. 2. An illustration of the integrated framework of KDGCN-IC and KDGCN-DC, where IC denotes initial residual connection, DC denotes dense connection, and HC denotes holistic connection. The process of recommending i_7 to u_1 is shown in figure.

integrated model framework of the two models, as shown in Fig. 2, is composed of three main parts: (1) the embedding layer, which initializes each node in $\boldsymbol{G}$ with a low-dimensional dense vector; (2) the aggregation layers, which iteratively propagate a node's neighbor embeddings as well as injecting its initial embedding or preceding layers' embeddings to update the embedding of itself, and generate its final embedding by combining the embeddings from all the aggregation layers; and (3) the prediction layer, which outputs a preference score of a user towards an item for the recommendation.

4.1. Embedding layer

To learn the embedding of users and items, we assign each node in $\boldsymbol{G}$ an initial embedding. To alleviate the dimension explosion of one-hot encoding, we parameterize the nodes with low-dimensional dense vectors

$$\mathbf{E} = \{\underbrace{\mathbf{e}_{u_1}, \mathbf{e}_{u_2}, \dots, \mathbf{e}_{u_M}}_{\text{users embeddings}}, \underbrace{\mathbf{e}_{i_1}, \mathbf{e}_{i_2}, \dots, \mathbf{e}_{i_N}}_{\text{items embeddings}}, \underbrace{\mathbf{e}_{e_{N+1}}, \mathbf{e}_{e_{N+2}}, \dots, \mathbf{e}_{e_O}}_{\text{entities\items embeddings}}\}, \tag{1}$$

where $\mathbf{E} \in \mathbb{R}^{(M+O)\times d_0}$ is a parameter matrix for users and entities (including items), d_0 ($\ll (M+O)$ in general) is the initial embedding dimension. In our frameworks, we update the initial embeddings by propagating them on the collaborative heterogeneous graph and achieve prediction in an end-to-end fashion.

4.2. Deep graph convolutional networks for recommendation

Inspired by convolutional neural networks, GCNs intend to extract features at a node by recursively aggregating feature information from its neighbors [21,44]. Specifically, GCNs perform graph convolutional operations iteratively to aggregate the neighbor features as the new representation of the central node. Following the propagation rule, the embedding of user u and item i at the $(l+1)_{th}$ layer can be represented as

$$\begin{aligned} \mathbf{e}_u^{(l+1)} &= \sigma\Big(\sum_{q\in\mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_q|}} \mathbf{e}_q^{(l)} \boldsymbol{W}^{(l)}\Big), \\ \mathbf{e}_i^{(l+1)} &= \sigma\Big(\sum_{v\in\mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i||\mathcal{N}_v|}} \mathbf{e}_v^{(l)} \boldsymbol{W}^{(l)}\Big), \end{aligned} \tag{2}$$

where $\mathcal{N}_u$ is the neighbor set of user u, and $|\mathcal{N}_u|$ is its cardinality, and $\frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_q|}}$ is the symmetric normalization term of the adjacency matrix of $\boldsymbol{G}$ at position (u,p); $\boldsymbol{W}^{(l)} \in \mathbb{R}^{d_l \times d_{l+1}}$ is a layer-wise trainable feature transformation matrix, shared by all nodes, where d_l is the hidden size of layer l; $\sigma(\cdot)$ is a nonlinear activation function, such as $ReLU(\cdot) = max(0,\cdot)$ [9]. $\mathbf{e}_u^{(0)} = \mathbf{e}_u$ and $\mathbf{e}_i^{(0)} = \mathbf{e}_i$ are the initial embedding of user u and item i, respectively. Note that in the collaborative heterogeneous graph $\boldsymbol{G}$, the neighbors of users are items and the neighbors of items can be users or non-item entities.

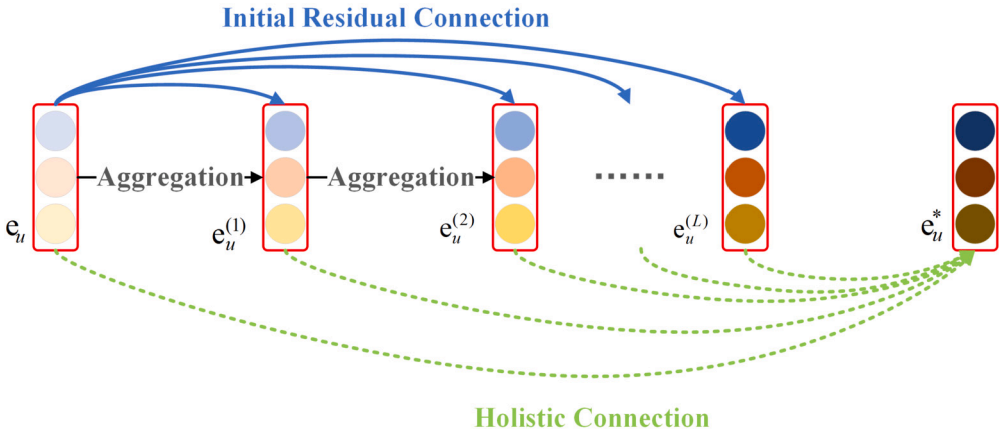


Fig. 3. An illustration of the graph neural network for the recommendation which contains the initial residual connection (blue solid lines) and the holistic connection (green dotted lines). The feature aggregation process of user u is shown in the figure.

Although the performance of GCN-based models has been greatly improved, they still suffer from over-smoothing and achieve their optimal performance in the shallow layers only. Therefore, these models are unable to capture higher-order features and accurately recommend higher-order items to users. To improve the ability to explore higher-order information and recommend higher-order items, we designed two deep GCN-based models to help alleviate the impact of over-smoothing.

4.2.1. *Initial residual connection in GCNs*

A vanilla GCN with K convolutional layers approximates a K_{th} order Laplacians polynomial filter with predetermined coefficients [40]. However, such a filter with fixed coefficients essentially simulates a lazy random walk, which limits its ability to express higher-order features and leads to over-smoothing [34]. Inspired by GCNII [6], we introduce an initial residual connection and identity mapping, which enable GCN to simulate a K_{th} order polynomial filter with arbitrary coefficients to alleviate over-smoothing and deepen the GCN. The embedding of user u and item i at the $(l+1)_{th}$ layer can be represented as

$$\begin{aligned}\mathbf{e}_u^{(l+1)} &= \sigma((\sum_{q\in\mathcal{N}_u}\frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_q|}}(1-\alpha_l)\mathbf{e}_q^{(l)}+\alpha_l\mathbf{e}_u^{(0)})((1-\beta_l)I_l+\beta_l\boldsymbol{W}^{(l)})),\\ \mathbf{e}_i^{(l+1)} &= \sigma((\sum_{v\in\mathcal{N}_i}\frac{1}{\sqrt{|\mathcal{N}_i||\mathcal{N}_v|}}(1-\alpha_l)\mathbf{e}_v^{(l)}+\alpha_l\mathbf{e}_i^{(0)})((1-\beta_l)I_l+\beta_l\boldsymbol{W}^{(l)})),\end{aligned} \tag{3}$$

where the term $+\alpha_l\mathbf{e}_u^{(0)}$ is the initial residual connection, which ensures that the final embedding of each node retains a part of its initial embedding to preserve locality; the term $(1-\beta_l)I_l$ is the identify mapping, which ensures that the deep GCN model can achieve performance similar to its shallow variant, where I_l is an identity matrix. Theoretically, both of the initial residual connection and identify mapping are conducive to avoiding over-smoothing. α_l and β_l are the two parameters to be tuned manually or trained automatically. Following GCNII [6], α_l is simply set to 0.1 or 0.2, and β_l is set to $\frac{\delta}{l}$, where δ is a hyperparameter. The non-item entities follow the same aggregation process to generate their final embeddings.

Previous works [13,5,33] have experimentally proven that feature transformation and nonlinear activation hurt the recommendation as well. Therefore, we simplified the feature aggregation process by removing the two components. Our proposed GCN with the initial residual connection is shown in Fig. 3, the feature aggregation process of which can be reformulated as

$$\begin{aligned}\mathbf{e}_u^{(l+1)} &= \sum_{q\in\mathcal{N}_u}\frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_q|}}(1-\alpha_l)\mathbf{e}_q^{(l)}+\alpha_l\mathbf{e}_u^{(0)},\\ \mathbf{e}_i^{(l+1)} &= \sum_{v\in\mathcal{N}_i}\frac{1}{\sqrt{|\mathcal{N}_i||\mathcal{N}_v|}}(1-\alpha_l)\mathbf{e}_v^{(l)}+\alpha_l\mathbf{e}_i^{(0)}.\end{aligned} \tag{4}$$

Note that when the feature transformation matrix $\boldsymbol{W}^{(l)}$ is removed, the combination of features and identity mapping is equivalent to multiplying the features by a constant, and a similar effect can be achieved by adjusting α_l. Therefore, we keep only the initial residual connection and discard the identity mapping. After L convolutional layers of feature aggregation, $L+1$ embeddings are gained for user u (item i), which represent the input embedding of u (item i) from the first layer (i.e., $\mathbf{e}_u$ ($\mathbf{e}_i$)) to the $(L+1)_{th}$ layer (i.e., $\mathbf{e}_u^{(L)}$ ($\mathbf{e}_i^{(L)}$)). Different from using the last layer's output embedding of each node for recommendation as do most models [47,37,36], we apply a layer combination, namely a holistic connection, to combine the $L+1$ embeddings to constitute the final embedding of each node. The final embedding is represented as

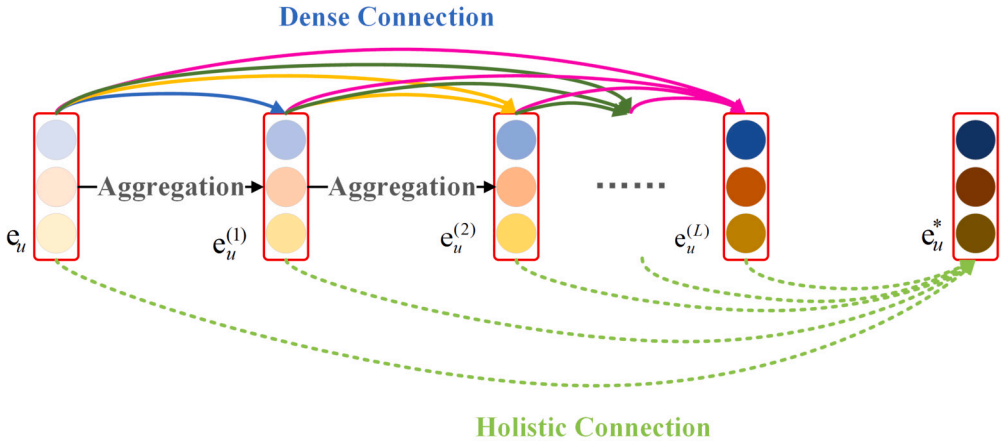


Fig. 4. An illustration of the graph neural network for the recommendation which contains the dense connection (solid lines) and the holistic connection (green dotted lines). The feature aggregation process of user u is shown in the figure.

$$\begin{aligned} \mathbf{e}_u^* &= \sum_{l=0}^{L} \gamma_l \mathbf{e}_u^{(l)}, \\ \mathbf{e}_i^* &= \sum_{l=0}^{L} \gamma_l \mathbf{e}_i^{(l)}, \end{aligned} \tag{5}$$

where $\gamma_l \geq 0$ is the weight of the l_{th} layer embedding in generating the final embedding, which is experimentally set to $\frac{1}{L+1}$ (i.e., the final embedding is the element-wise mean aggregation of the former embeddings). Since the holistic connection takes the same effect as self-connections, we only aggregate the neighbor features in the feature aggregation process without integrating the feature of the target node.

4.2.2. Dense connection in GCNs

DenseNet [15] improves the information flow between the layers by directly connecting each layer to its preceding layers in a feed-forward manner to encourage sufficient and effective feature reuse. DenseGCN [21] deepens GCN by following the network architecture of DenseNet and achieves significant progress in the point cloud semantic segmentation task. Following DenseGCN, we propose a GCN with a dense connection and holistic connection, as shown in Fig. 4. The feature aggregation process of user u and item i at the $(l+1)_{th}$ layer can be formulated as

$$\begin{aligned} \mathbf{e}_u^{(l+1)} &= \sum_{q \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_q|}} (1-\alpha_l)\mathbf{e}_q^{(l)} + \alpha_l AGG(\mathbf{e}_u, \mathbf{e}_u^{(1)}, \ldots, \mathbf{e}_u^{(l)}), \\ \mathbf{e}_i^{(l+1)} &= \sum_{v \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i||\mathcal{N}_v|}} (1-\alpha_l)\mathbf{e}_v^{(l)} + \alpha_l AGG(\mathbf{e}_i, \mathbf{e}_i^{(1)}, \ldots, \mathbf{e}_i^{(l)}), \end{aligned} \tag{6}$$

where $AGG(\cdot)$ is an aggregation function which combines the preceding layers' embeddings of the target node into an integrated embedding, formulated as:

$$\begin{aligned} AGG(\mathbf{e}_u, \mathbf{e}_u^{(1)}, \ldots, \mathbf{e}_u^{(l)}) &= \sum_{h=0}^{l} \frac{1}{1+exp(h)} \mathbf{e}_u^{(h)}, \\ AGG(\mathbf{e}_i, \mathbf{e}_i^{(1)}, \ldots, \mathbf{e}_i^{(l)}) &= \sum_{h=0}^{l} \frac{1}{1+exp(h)} \mathbf{e}_i^{(h)}. \end{aligned} \tag{7}$$

A layer sensitivity weighted sum aggregator is used in our model. Following the thought of GCNII [6], we make the contribution negatively correlated to the layer (i.e., the embedding of the lower layer contributes more to the embedding generation of the subsequent layer). We leave the exploration of other types of aggregators, such as weighted mean pooling, max pooling, and LSTM for future work. After L layers' aggregation, the final embedding of user u and item i are obtained through the holistic connection, i.e., Eq. (5).

4.3. Model prediction

After obtaining the final embedding of user u and item i, a function $f : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ is used to output the probability that user u has a potential preference towards item i. The probability is used as the metric for the recommendation. In this work, we employed the simple similarity measure function of the inner product

$$\hat{y}_{ui} = \mathbf{e}_u^{*T} \mathbf{e}_i^*. \tag{8}$$

4.4. Optimization

We optimize the trainable parameters by minimizing the *Bayesian Personalized Ranking* [30] (BPR) loss, which is a pairwise loss that encourages a user to show more preference in their observed items than the unobserved items

$$\mathcal{L} = -\sum_{(u,i,j)\in\mathcal{O}} \ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda \|\theta\|_2^2, \tag{9}$$

where $\mathcal{O} = \{(u,i,j)|(u,i) \in \mathcal{O}^+, (u,j) \in \mathcal{O}^-\}$ denotes the pairwise training dataset, in which $\mathcal{O}^+$ indicates the positive observed user-item pairs and $\mathcal{O}^-$ indicates the sampled unobserved negative user-item pairs; $\sigma(\cdot) = (1 + exp(-\cdot))^{-1}$ is the logistic sigmoid; λ is the penalty strength of the L_2 regularization that is introduced to prevent overfitting; and $\theta = \mathbf{E}$ is the trainable parameters. The mini-batch Adam [16] optimizer is adopted to refine the trainable parameters due to its effectiveness in preventing local optima.

4.5. Model analysis

Model size. In terms of the model size, all the parameters of KDGCN-IC and KDGCN-DC come from the node embeddings (i.e., $\theta = \mathbf{E}$), which are heavier than that of LightGCN [13] and SGL-ED [41] (i.e., $\theta = \{\mathbf{E}_u, \mathbf{E}_i\}$, in which $\mathbf{E}_u$ and $\mathbf{E}_i$ are the initial embeddings of users and items, respectively), and slightly lighter than that of KGAT [37] (i.e., $\theta = \{\mathbf{E}, \sum_{r=0}^{\mathcal{R}} W_r, \sum_{l=0}^{L} W^{(l)}\}$, in which $W_r \in \mathbb{R}^{d_0 \times d_1}$ is the transformation matrix of relation r in knowledge graph embedding). However, since our models do not introduce additional trainable parameters beyond the initial embeddings, the aggregation layer is lightweight, which is identical to LightGCN and SGL-ED, and lighter than NGCF [38] and KGAT. Thus, compared to LightGCN and SGL-ED, we introduce the embedding of the non-item entities, while compared to KGAT, we omit the additional feature transformation matrix. For example, on our experimented Book-Crossing dataset (with 18 thousand users, 15 thousand items, 25 relations, and 63 thousand non-item entities), when the embedding size is 64, and we use 3 aggregation layers for NGCF of size 64×64, and a tower-structure of two layers for KGAT (i.e., $64 \rightarrow 32 \rightarrow 16$), the total number of parameters of LightGCN and SGL-ED is 2.10 million, and that of NGCF, our models, and KGAT are 2.12 million, 6.13 million, and 6.24 million, respectively. However, there are 12.23 thousand and 107.52 thousand additional parameters for NGCF and KGAT, which may increase the difficulty of model training. To summarize, KDGCN-IC and KDGCN-DC pursue a balance of model size and accuracy by introducing item knowledge graphs and discarding additional weight matrices.

Model complexity. Since the time complexity of the initial residual connection of KDGCN-IC and the dense connection of KDGCN-DC is a constant, the time complexity of KDGCN-IC and KDGCN-DC is the same. Suppose the number of nodes and edges in the collaborative heterogeneous graph $\boldsymbol{G}$ are $|V|$ and $|E|$ respectively, the number of training epochs is p, the size of each training batch is B. In addition, d_0 is the size of the initial embedding, and L denotes the number of aggregation layers. The time cost comes from three parts. For the symmetric normalization of adjacency matrix, since the number of non-zero elements in the adjacency matrix of $\boldsymbol{G}$ is $2|E|$, its time complexity is $\mathcal{O}(2|E|)$. For the graph convolution part, within a mini-batch, the time cost is $\mathcal{O}(2|E|d_0L)$, therefore, the time cost of the whole training is $\mathcal{O}(2|E|d_0Lp\frac{|E|}{B})$. For the prediction layer, only the inner product is conducted when calculating the BPR loss, the time cost of which is $\mathcal{O}(2|E|d_0p)$. In summary, the overall training complexity of KDGCN-IC and KDGCN-DC is $\mathcal{O}(2|E| + 2|E|d_0Lp\frac{|E|}{B} + 2|E|d_0p)$.

In practice, taking the Book-Crossing dataset as an example, the time cost of each epoch is 1.3 s, 0.7 s, 0.7 s, 1.1 s, 2.0 s, 5.8 s, 18.0 min, 3.2 s, 3.2 s for NGCF, LightGCN, DGCN-HN [11], SGL-ED, CKE [48], KGAT, CKAN [39], KGDCN-IC, and KDGCN-DC respectively. As we can see, even though the number of layers is much higher than the existing GCN-based model (NGCF, LightGCN, DGCN-HN, and SGL-ED) and knowledge graph-aware model (KGAT and CKAN) (as shown in Table 5), KDGCN-IC and KDGCN-DC are even more efficient than KGAT and CKAN.

4.6. Model discussion

To distinguish our work from that of previous researchers, we compare our models to eight closely related models in seven key characteristics, as shown in Table 3. First, we combined the user-item interaction graph and the item knowledge graph to bring similar items closer and to enable the users to see unconnected items that might be of interest, as does KGAT. In this way, the final embedding of the users and items contains sufficient collaborative information and semantic information to facilitate recommendations. Second, LR-GCCF and LightGCN proved the redundancy of the feature transformation matrix and nonlinear activation function, and also UltraGCN omitted these two components; therefore, we also removed them to reduce the trainable parameters and simplify the model training. Third, the holistic connection was applied to address the over-smoothing in NGCF and LightGCN, however, this technique can only alleviate the over-smoothing to a certain extent, so it is difficult for them to deepen the models to more than four layers. Fourth, DGCN-HN combines the holistic connection and residual connection to deepen GCN, however, it can only deepen up to ten layers, and moreover, we are more concerned with the initial representations. We experimentally introduced an initial residual connection and a dense connection to reuse previous features to deepen models. The dense connection can be considered as a variant of the initial residual connection because the initial features are also passed to each layer.

Table 3
Comparison of different GCN-based recommendation models. Use User-Item Graph indicates explicitly encoding the user-item interactions rather than just defining them in the loss function.

	Use U-I Graph	Use KG	Remove FT	Remove NA	Use IC	Use DC	Use HC
LR-GCCF [5]	√	×	×	√	×	×	×
NGCF [38]	√	×	×	×	×	×	√
LightGCN [13]	√	×	√	√	×	×	√
UltraGCN [28]	√	×	√	√	×	×	×
KGCN [36]	×	√	×	×	×	×	×
KGNNLS [35]	×	√	×	×	×	×	×
KGAT [37]	√	√	×	×	×	×	×
DGCN-HN [11]	√	×	√	√	×	×	√
KDGCN-IC	√	√	√	√	√	×	√
KDGCN-DC	√	√	√	√	√	√	√

[1] U-I graph means user-item graph.
[2] KG means item knowledge graph.
[3] FT means feature transformation.
[4] NA means nonlinear activation.
[5] IC means initial residual connection.
[6] DC means dense connection.
[7] HC means holistic connection.

Table 4
Statistics of datasets. KG means item knowledge graph.

	Book-Crossing	MovieLens-20M	Last-fm
users	17857	3988	1872
items	14967	16703	3846
interactions	58782	199629	21173
sparsity	0.0002	0.0030	0.0029
entities	77903	102569	9366
relations	25	32	60
KG triples	151500	499474	15518

5. Experiments

5.1. Dataset description

We verified the effectiveness of KDGCN-IC and KDGCN-DC by conducting extensive experiments on three benchmark real-world datasets: Book-Crossing, MovieLens-20M, and Last-fm. These datasets are public, widely used in academic research, and vary in size and sparsity.

- Book-Crossing contains nearly 1.15 million explicit rating records of books in the Book-Crossing community.
- MovieLens-20M contains almost 20 million explicit rating records of movies on the MovieLens website.
- Last-fm contains approximately 93 thousand explicit rating records of music from Last.fm online music systems.

Since these datasets are explicit feedback, we converted them into implicit feedback. Each user-item pair with a score not lower than a threshold is considered as a positive sample. We set the threshold for MovieLens-20M to four, while no thresholds were set for the other two datasets due to their sparsity.

In addition to the implicit feedback, we also needed the corresponding item knowledge graph of each dataset. The three datasets and their corresponding knowledge graphs were provided by [36]. We only used the first 200 thousand positive records in MovieLens-20M to reduce the time cost, and we dropped the users with fewer than five interactions in the three datasets to ensure the quality of the datasets. Table 4 summarizes the statistics of these datasets after pre-treating.

We followed the same strategy as NGCF [38] to split the datasets and sample the negative samples: For each dataset, 80% of the historical records of each user were randomly selected to constitute the training set, and the remaining records were used as the test set. From the training set, we randomly sampled 10% of the records as the validation set for hyperparameter tuning. We treated each user-item pair in the training set as a positive instance and paired the user with a negative item that was not observed to interact with the user in the training set.

5.2. Experimental settings

5.2.1. Evaluation metrics

We adopted two widely-used evaluation metrics, Recall@$\mathcal{K}$ and NDCG@$\mathcal{K}$, to evaluate the effectiveness of the models on the top-$\mathcal{K}$ recommendation and user preference ranking. The larger the metric value, the better the performance. $\mathcal{K} = 20$ by default.

- Recall@$\mathcal{K}$ is the proportion of the positive items successfully recommended in the top-$\mathcal{K}$ items, which only considers the relative position of the recommended items. The average recall of all users is reported as the final recall.

$$Recall@\mathcal{K} = \sum_{u \in \boldsymbol{U}_t} \frac{|Rel_u \cap Rec_u|}{N_t * |Rel_u|}, \tag{10}$$

where Rel_u is the interacted items of user u in the test set; Rec_u is the top-$\mathcal{K}$ items in the recommended item list; $\boldsymbol{U}_t$ is the user set, and N_t is its number.

- Normalized Discounted Cumulative Gain@$\mathcal{K}$ (NDCG@$\mathcal{K}$) is a standard metric in recommender systems that assigns higher scores to the positive items at higher positions.

$$\begin{aligned} NDCG@\mathcal{K} &= \sum_{u \in \boldsymbol{U}_t} \frac{DCG_u@\mathcal{K}}{N_t * IDCG_u@\mathcal{K}}, \\ DCG_u@\mathcal{K} &= \sum_{p=1}^{\mathcal{K}} \frac{2^{rel_p} - 1}{log_2(p+1)}, \\ IDCG_u@\mathcal{K} &= \sum_{p=1}^{\mathcal{K}} \frac{2^{\hat{rel}_p} - 1}{log_2(p+1)}, \end{aligned} \tag{11}$$

where $rel_p \in \{0,1\}$ and $\hat{rel}_p \in \{0,1\}$ are the relationships between the recommended items in position p and the target user. $DCG_u@\mathcal{K}$ reflects the score of the actual recommendation list of user u, and $IDCG_u@\mathcal{K}$ reflects the score of the ideal recommendation list of user u.

5.2.2. Baselines

To evaluate the effectiveness of KDGCN-IC and KDGCN-DC, we compared them to several methods, including GCN-based models NGCF [38], LightGCN [13], and DGCN-HN [11], self-supervised learning model SGL-ED [41], and knowledge graph-aware models CKE [48], KGAT [37], and CKAN [39]. The well-known baseline MF [20] and NeuMF [14] were not been compared because baselines, like LightGCN and KGAT, have been shown to outperform them. Also, KGCN [36] has been shown to underperform KGAT and CKAN and is hard to implement in deep layers. All the models were implemented in the environment of NeuRec, except for CKAN, on whose source code the experiments were performed. The details of the comparison models are as follows:

- NGCF [38] is a state-of-the-art GCN-based model that propagates the features in the user-item bipartite graph via a standard multi-layer GCN to capture the high-order collaborative signal in the graph. The concatenation of the embedding at each layer is used as the final embedding.
- LightGCN [13] improves NGCF by omitting the feature transformation matrix and nonlinear activation function, which are two standard components in GCN but redundant for CF. The holistic connection is utilized to obtain the final embedding.
- DGCN-HN [11] introduces the residual connection on the basis of LightGCN to model the higher-order collaborative signals between users and items, as well as a hybrid normalization to aggregate different neighbor information. We implemented the hybrid normalization as the average of symmetric normalization and left normalization.
- SGL-ED [41] assists the training of LightGCN by employing edge dropout [31] to generate multiple views for each node in the user-item graph and maximize the consistency of different views of the same node.
- CKE [48] enhances item representations based on MF by exploiting semantic representations from structural knowledge, textual knowledge, and visual knowledge. Here, we only used the structural knowledge (i.e., item knowledge graph), and encoded it with TransR [25].
- KGAT [37] applies GCN to the hybrid graph of the user-item graph and item knowledge graph to learn the node representation and uses TransR to refine the representation.
- CKAN [39] explicitly encodes the information in the user-item graph and the item knowledge graph through a heterogeneous information propagation strategy, and employs an attention mechanism to discriminate the importance of different neighbors.

5.2.3. Parameter settings

For all the models, the embedding size and batch size were set to 64 and 1024, respectively; and the trainable parameters were initialized with the Xavier initialization methods [8]. We carefully search the best setting for the following hyper-parameters: the learning rate was searched in $\{10^{-2}, 5\times10^{-3}, 10^{-3}, 5\times10^{-4}, 10^{-4}\}$; the strength coefficient of L_2 normalization λ was tuned in $\{10^{-6}, 10^{-5}, \cdots, 10^{-1}\}$; the number of graph convolutional layers was tuned in the range of $\{1,2,3,4\}$ for NGCF, LightGCN, SGL-ED, KGAT, and CKAN, and was searched in the range of $\{1,2,\cdots,10\}$ for DGCN-HN, and was searched in $\{1,2,3,4,8,16,32,64\}$ for our models; the edge dropout ratio of SGL-ED was tuned among $\{0.1,0.2,0.3,0.4,0.5\}$, and the temperature of which was searched in $\{0.1,0.2,0.5,1.0\}$; and the size of triple for users and items was tuned in $\{4,8,16,32,64\}$ for CKAN. For KGAT, the hidden dimension of the first layer was 64, and that of each subsequent layer was half of its previous layer. For NGCF, we designed a tower structure similar to KGAT or fixed the hidden dimension to the embedding size. For NGCF and KGAT, the node dropout and message dropout techniques were employed, and their drop ratios were set to 0.1. In addition, we adopted an early stopping strategy for all the models, i.e., stop training prematurely if Recall@$\mathcal{K}$ on the validation set does not increase for 50 consecutive epochs.

Table 5
Overall comparison with baseline methods on Book-Crossing, MovieLens-20M, and Last-fm. The numbers in parentheses correspond to the model depths.

	Book-Crossing		MovieLens-20M		Last-fm	
	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
NGCF	0.1102(3)	0.0604(3)	0.3353(3)	0.2664(3)	0.4046(2)	0.2286(2)
LightGCN	0.1291(3)	0.0722(3)	0.3360(2)	0.2637(2)	0.4092(2)	0.2395(2)
DGCN-HN	0.1285(3)	0.0751(3)	0.3403(8)	0.2703(8)	0.4130(4)	0.2355(4)
SGL-ED	0.1311(4)	0.0747(4)	0.3304(2)	0.2580(2)	0.4119(2)	**0.2396**(2)
CKE	0.1049(-)	0.0558(-)	0.3258(-)	0.2534(-)	0.3863(-)	0.2213(-)
KGAT	0.1145(2)	0.0659(2)	0.3397(2)	0.2692(2)	0.4002(2)	0.2278(2)
CKAN	0.1056(2)	0.0664(2)	0.2926(2)	0.2286(2)	0.3692(2)	0.2121(2)
KDGCN-IC	0.1540(32)	**0.0861**(32)	0.3545(32)	0.2832(32)	**0.4168**(16)	0.2389(16)
KDGCN-DC	**0.1543**(32)	0.0860(32)	**0.3561**(8)	**0.2835**(8)	0.4150(8)	0.2377(8)

* CKE is not a GCN-based model.

5.3. Overall comparison

Table 5 shows the performance comparison between KDGCN-IC, KDGCN-DC, and the variant baselines. We concluded the following:

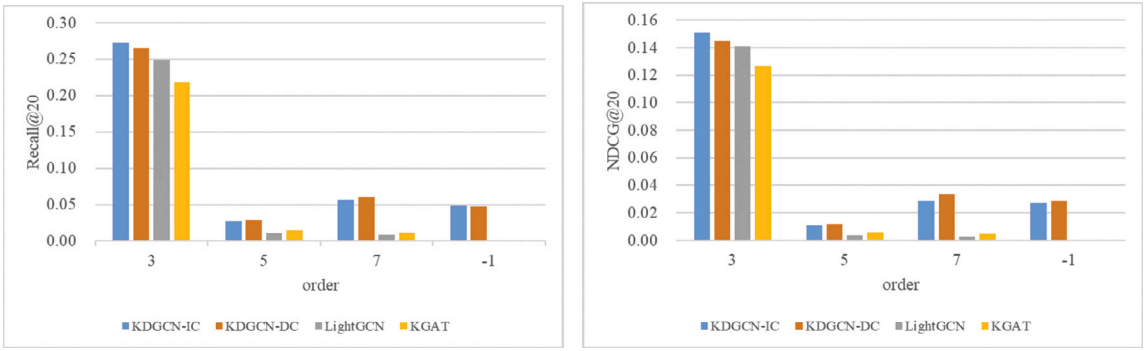
- KDGCN-IC and KDGCN-DC consistently achieved the best performance on the three datasets, except that NDCG@20 on Last-fm was slightly lower than LightGCN and SGL-ED. In particular, our models improved over the best performance baselines *w.r.t.* Recall@20 by 15.67%, 4.64%, and 0.92% on Book-Crossing, MovieLens-20M, and Last-fm, respectively. By introducing the item knowledge graph, our models were able to reconstruct the connectivity between the nodes and make the connectivity more reasonable. By improving the information flow, KDGCN-IC and KDGCN-DC were able to model the higher-order connectivity simply and effectively, thereby enabling sufficient collaborative signals to flow into the node embedding. Compared with KDGCN-IC, KDGCN-DC achieved comparable performance to KDGCN-IC at shallower layers on MovieLens-20M and Last-fm datasets, reducing the computational cost of model training. Moreover, we emphasize that almost all the user-item pairs in the MovieLens-20M test set belong to ord_3, indicating that our models also had a certain positive effect on the low-order recommendation.
- Compared with NGCF, LightGCN achieved significant improvement on the three datasets, which indicates that removing the feature transformation matrix and nonlinear activation function was beneficial for the recommendation. KGAT performed better than NGCF on Book-Crossing and MovieLens-20M, which was a reasonable outcome because KGAT injected sufficient semantic information into the final representations by introducing the item knowledge graph as well as the auxiliary training of TransR.
- Compared with LightGCN, SGL-ED achieved better performance on Book-Crossing and Last-fm, and even NDGC@20 on Last-fm was better than KDGCN-IC and KDGCN-DC, which may be due to the fact that the data augmentation helped the network to learn multi-view structural features, thus reinforcing the node representation. DGCN-HN outperformed LightGCN on MovieLens-20M and Last-fm but underperformed on Book-Crossing, which verified the effectiveness of aggregating higher-order information.
- Compared with KGAT, LightGCN performed better in most cases. Although KGAT enriched the node representation through the item knowledge graph, it introduced numerous trainable parameters while retaining the complex calculation process (i.e., the feature transformation and nonlinear activation), which increased the training difficulty.
- Compared the knowledge graph-aware methods, CKAN performed worst on MovieLens-20M and Last-fm datasets, possibly owing to the imbalance of neighbor feature aggregation brought about by simple random sampling. In addition, the performance of CKE was consistently worse than KGAT, which indicates that the learned embeddings via the knowledge graph embedding task may be unsuitable for the recommendation, and a better solution would be to conduct the recommendation task with the aid of the knowledge graph embedding task (i.e., using TransR to fine-tune the node embeddings).

5.4. Ablation and effectiveness analysis

5.4.1. Performance comparison w.r.t. high-order recommendation

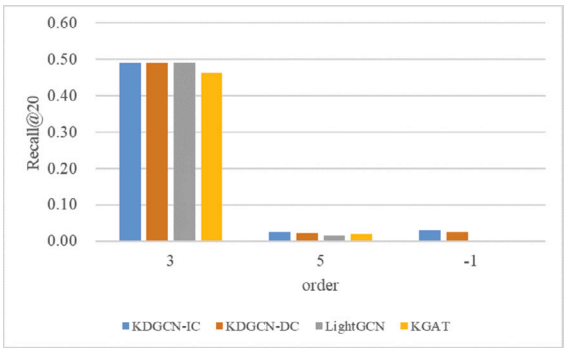
The major goal of our work was to boost the model performance in recommending high-order items. We compared KDGCN-IC and KDGCN-DC with LightGCN and KGAT in recommending user-item pairs with different orders. Fig. 5 shows the results on the Book-Crossing and Last-fm datasets. We concluded the following:

- LightGCN demonstrated obvious shortcomings in recommending high-order items. As the order increased, LightGCN became less effective, this was especially true when recommending items that were not connected to the target users where it was almost impossible for LightGCN to achieve accurate predictions. For example, the Recall@20 on ord_7 and ord_-1 of Book-Crossing were 0.0082 and 0.0006, respectively, and that of Last-fm were 0.0161 and 0.0000, respectively.
- Compared with LightGCN, the higher-order recommendation accuracy of KGAT was improved, but its accuracy of lower-order recommendation was significantly reduced, which indicated that introducing an item knowledge graph to reconstruct the con-

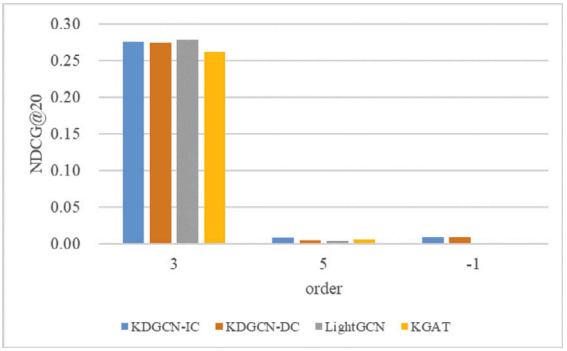


(a) Recall@20 on Book-Crossing

(b) NDCG@20 on Book-Crossing



(c) Recall@20 on Last-fm



(d) NDCG@20 on Last-fm

Fig. 5. Performance comparison *w.r.t.* different orders between items and the target user.

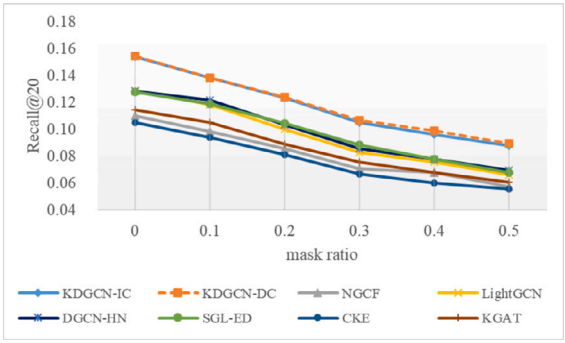
nectivity of users and items in order to improve higher-order recommendation capability of the model was effective but not sufficient.

- Compared with LightGCN, KDGCN-IC and KDGCN-DC significantly improved the recommendation accuracy of high-order items without almost losing the recommendation accuracy of the low-order items, for example, the Recall@20 of KDGCN-IC and KDGCN-DC on ord_-1 of Book-Crossing were 0.0481 and 0.0473 respectively, and that of Last-fm were 0.0294 and 0.0261 respectively, which verified the effectiveness of simultaneously addressing the over-smoothing and lack of connectivity issues to improve the high-order recommendation capability. In addition, compared to KDGCN-IC, KDGCN-DC showed comparable or even better performance on the higher-order recommendation, while KDGCN-IC was more advantageous on the lower-order recommendation.

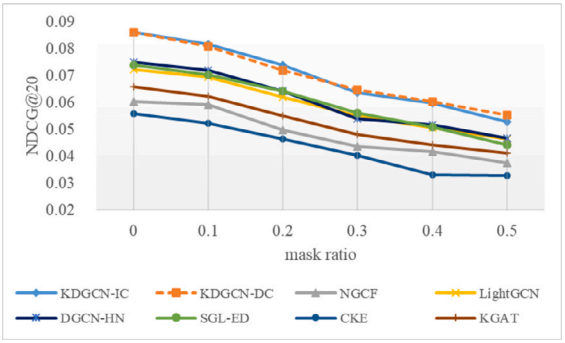
5.4.2. Performance comparison w.r.t. cold-start scenarios

One of main purposes for introducing the item knowledge graph to recommendation was to solve the cold-start issue. To verify the effectiveness of KDGCN-IC and KDGCN-DC in cold-start scenarios, we randomly masked a certain ratio of items in the training set while keeping the test set unchanged. The mask ratio varied within $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. The results on the Book-Crossing, MovieLens-20M, and Last-fm datasets are shown in Fig. 6. It is worth noting that we did not compare CKAN because it requires every user in the test set to appear in the training set, which is difficult to guarantee when the mask ratio is high. We concluded the following:

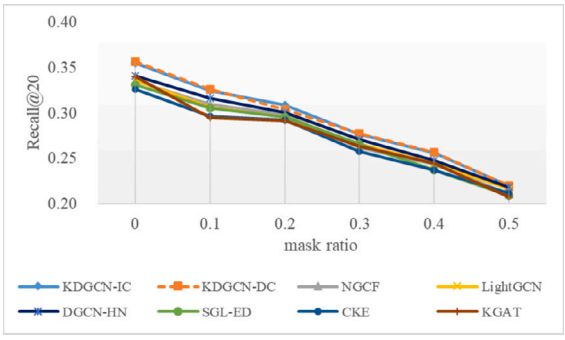
- KDGCN-IC and KDGCN-DC maintained their top-two performance in most cases. Specifically, when the mask ratio was 0.5, Recall@20 on Book-Crossing decreased by 47.91%, 48.57%, 45.84%, 47.15%, 47.00% and 46.90% for the six baselines compared to the performance on the full training set (i.e., mask ratio was 0), and the reductions of KDGCN-IC and KDGCN-DC were 42.92% and 42.00% respectively, which were significantly lower than the baselines. Moreover, as the mask ratio increased, KDGCN-IC and KDGCN-DC outperformed baselines on Last-fm by an increasing margin. The performance of KDGCN-IC and KDGCN-DC indicated that they maintained significant predictive ability for the cold-start scenarios. In particular, the KDGCN-DC achieved better performance compared to the KDGCN-IC in the case of extreme cold start (i.e., mask ratio greater than 0.3).
- In most cases, NGCF and KGAT underperformed LightGCN and SGL-ED, mainly because they contained a large number of trainable parameters and therefore increasing the dependence on the number of training samples. In addition, although the total number of trainable parameters of KDGCN-IC, KDGCN-DC and KGAT were comparable, KDGCN-IC and KDGCN-DC significantly



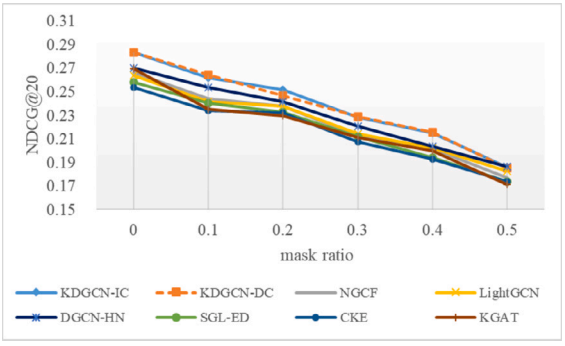
(a) Recall@20 on Book-Crossing



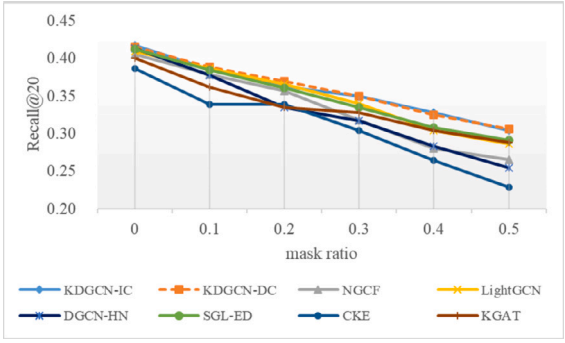
(b) NDCG@20 on Book-Crossing



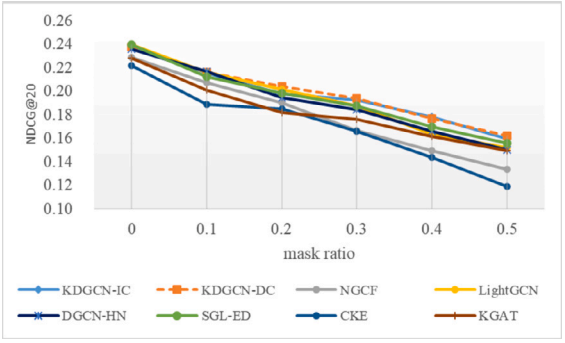
(c) Recall@20 on MovieLens-20M



(d) NDCG@20 on MovieLens-20M



(e) Recall@20 on Last-fm



(f) NDCG@20 on Last-fm

Fig. 6. Performance comparison *w.r.t.* different mask ratios of the training nodes.

outperformed KGAT in all cases, which may be due to the fact that they did not introduce additional weight matrices and were able to capture higher-order neighbor information.

5.4.3. Impact of proposed components

To investigate the effectiveness of the proposed components in alleviating the over-smoothing issue, we conducted an ablation experiment on the three datasets. Since the optimal values of the hyperparameters of KDGCN-IC and KDGCN-DC were the same on each dataset except the layer, we set the number of layers of both models on a dataset to the number at which KDGCN-DC achieved the best performance, and integrated the experimental results in a unified table. Table 6 presents the results on the three datasets. We concluded the following:

- The introduced four components of our models (initial residual connection, dense connection, holistic connection, and item knowledge graph) demonstrated their effectiveness in alleviating over-smoothing.
- Comparing the performance of KDGCN-IC, KDGCN-DC, KDGCN w/o C, KDGCN-IC w/o HC and KDGCN-DC w/o HC verified that the introduction of the initial residual connection, dense connection, and holistic connection components helped to prevent over-smoothing and improved recommendation performance. The main reason for this outcome may be the feature reuse capability.

Table 6
Ablation study of components.

	Book-Crossing		MovieLens-20M		Last-fm	
	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
KDGCN-IC	0.1540	0.0861	0.3483	0.2783	0.4135	0.2381
KDGCN-DC	0.1543	0.0860	0.3561	0.2835	0.4150	0.2377
KDGCN w/o C	0.1421	0.0787	0.2042	0.1633	0.3603	0.1979
KDGCN-IC w/o HC	0.1522	0.0853	0.3448	0.2736	0.4056	0.2255
KDGCN-DC w/o HC	0.1532	0.0856	0.3393	0.2703	0.4028	0.2267
KDGCN-IC w/o KG	0.1193	0.0681	0.3471	0.2764	0.4074	0.2305
KDGCN-DC w/o KG	0.1231	0.0705	0.3467	0.2761	0.4070	0.2310
KDGCN w/o C&HC	0.1079	0.0701	0.2025	0.1629	0.2512	0.1375
KDGCN w/o C&HC&KG	0.0983	0.0682	0.2034	0.1642	0.1886	0.0906

[1] KDGCN is the collective name of KDGCN-IC and KDGCN-DC.
[2] C is the collective name of initial residual connection and dense connection.
[3] HC means holistic connection.
[4] KG means item knowledge graph.
[5] w/o means remove operation.

- Comparing the performance of KDGCN-IC and KDGCN-IC w/o KG, KDGCN-DC and KDGCN-DC w/o KG, and KDGCN w/o C&HC and KDGCN w/o C&HC&KG verified that introducing the item knowledge graph led to significant improvement on Book-Crossing and Last-fm, proving that the item knowledge graph can alleviate over-smoothing to some extent. This result may have been due to the graph's ability to increase the diversity of the node features, which to a certain extent delays the convergence of the node representations to a fixed vector. However, the item knowledge graph and the holistic connection appeared to have little effect on MovieLens-20M because the degree of the node was relatively high, which accelerated the convergence of the node representations to a fixed vector.

5.4.4. Impact of network layers

To explore the impact of the model depth on the recommendation performance of different models, we varied the number of layers in $\{1, 2, 3, 4, 8, 16, 32, 64\}$. Fig. 7 shows the results of our evaluation metrics on Book-Crossing, MovieLens-20M, and Last-fm datasets, in which LightGCN-f is a variant of LightGCN that removes the holistic connection in LightGCN (i.e., the representation at the final layer is used for prediction); ResLightGCN combines the residual connection and LightGCN, whose feature aggregation process is defined as

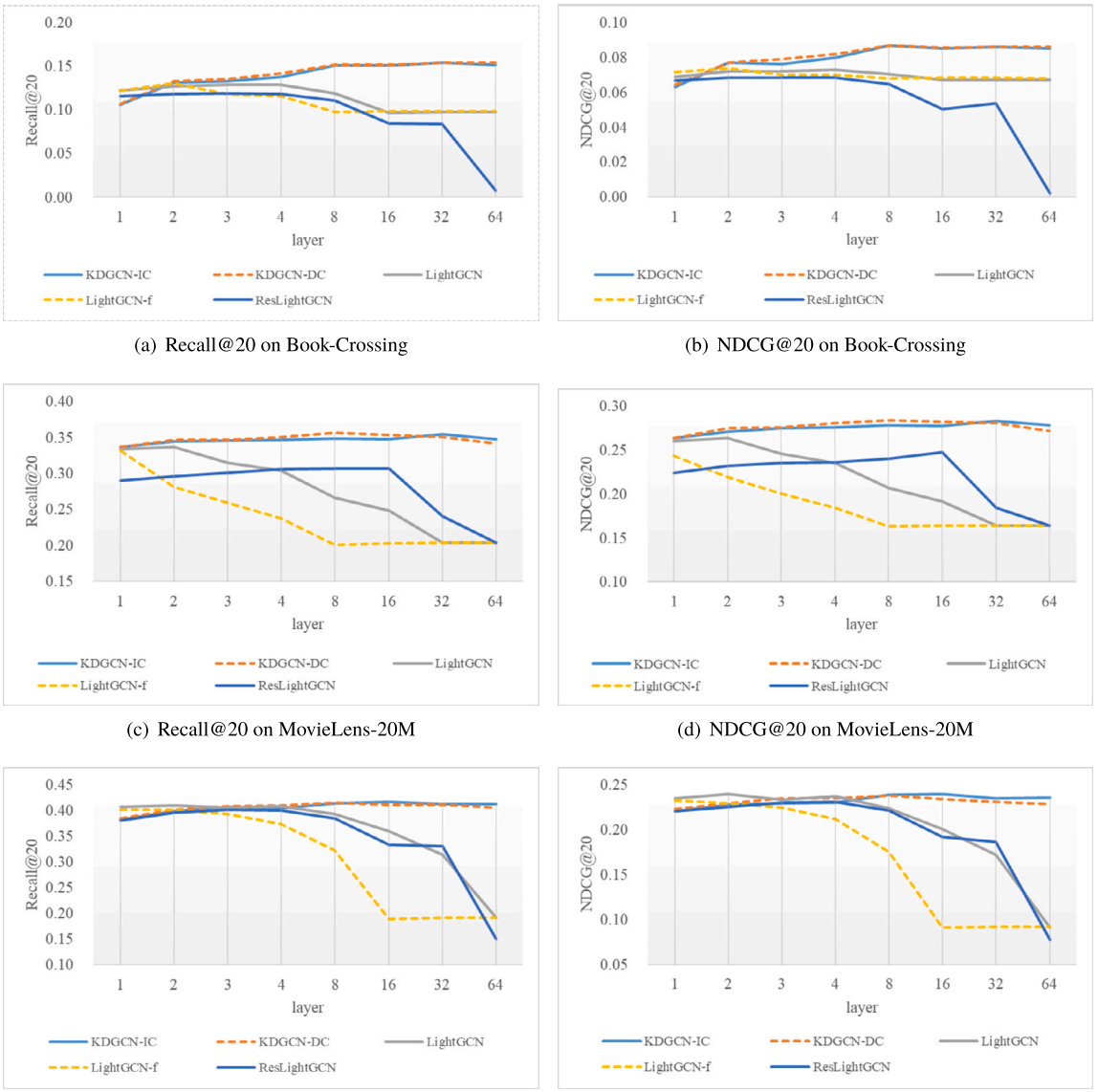
$$\begin{aligned} \mathbf{e}_u^{(l+1)} &= \sum_{q \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_q|}} \mathbf{e}_q^{(l)} + \mathbf{e}_u^{(l)}, \\ \mathbf{e}_i^{(l+1)} &= \sum_{v \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i||\mathcal{N}_v|}} \mathbf{e}_v^{(l)} + \mathbf{e}_i^{(l)}. \end{aligned} \tag{12}$$

We concluded the following:

- KDGCN-IC and KDGCN-DC well alleviated over-smoothing. LightGCN, LightGCN-f, and ResLightGCN consistently achieved their best performance at shallow layers (below four layers), except for ResLightGCN on the MovieLens-20M dataset where the best number of layers was 16. However, as the model deepened, the performance of KDGCN-IC and KDGCN-DC continued to rise and remained at a high level, which demonstrates the effectiveness of feature reuse in mitigating over-smoothing. Especially, in the shallow layer, KDGCN-DC performed better than KDGCN-IC; however, in the deep layer, KDGCN-IC outperformed KDGCN-DC, which indicates that the deep features played a more important role in the shallow model, while the shallow features were more important to the deep model.
- Comparing the performance curves of LightGCN and LightGCN-f verified that the holistic connection alleviated the over-smoothing problem to a certain extent. As the model deepened, the performance of LightGCN degraded more slowly than LightGCN-f, but it struggled to maintain a high level when the number of layers exceeded 4.
- The direct combination of a residual connection and LightGCN may not be suitable for our purposes. As the model deepened, the performance of ResLightGCN continued to decline, and the decreasing trend became increasingly intense, especially on the Book-Crossing and Last-fm datasets, which would have been due to the scale explosion caused by the direct addition of embeddings.

5.4.5. Impact of α_l

To explore the effect of the weight of previous representations, which greatly influences the generation of the final representation, we varied α_l in the range of $\{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$. Fig. 8 shows the result on all three datasets. We concluded the following:



(a) Recall@20 on Book-Crossing

(b) NDCG@20 on Book-Crossing

(c) Recall@20 on MovieLens-20M

(d) NDCG@20 on MovieLens-20M

(e) Recall@20 on Last-fm

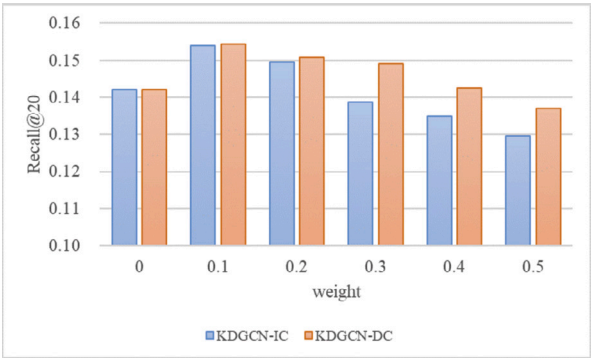
(f) NDCG@20 on Last-fm

Fig. 7. Performance comparison *w.r.t.* different model depths.

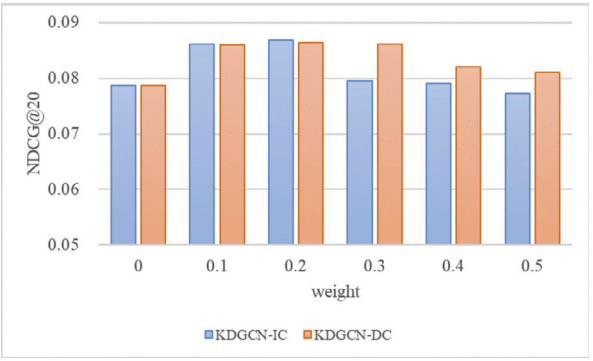
- With the gradual increase of α_l, the performance of KDGCN-IC and KDGCN-DC showed a noticeable trend of increasing and then decreasing, where they reached the optimal performance on the Book-Crossing, MovieLens-20M, and last-fm datasets when α_l was 0.1, 0.2, and 0.2 respectively, indicating that reusing a small percentage of features during representation generation facilitated the performance improvement.
- When the value of α_l was 0, KDGCN-IC and KDGCN-DC consistently performed poorly on all three datasets, while both models performed significantly better when α_l was 0.1 or 0.2, again demonstrating the contribution of initial residual connection and dense connection to model deepening. In addition, when α_l exceeded 0.2, the decline in performance of KDGCN-DC was slower than that of KDGCN-IC, suggesting that dense connection was more likely to help the model maintain performance at a deeper layer.

6. Conclusion

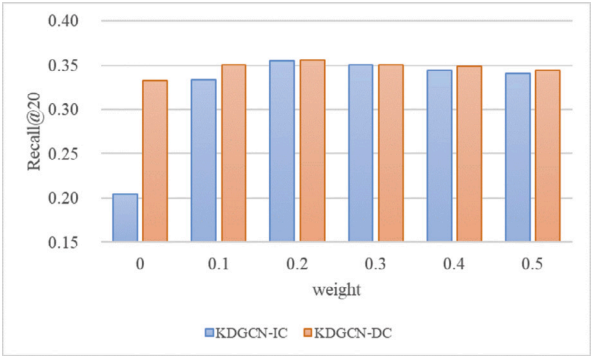
In this paper, we first discussed the shortcoming of modern GCN-based models in recommending higher-order items. Then, we focused on our efforts to address the shortcoming, mainly by connecting the cold-start items with the target user and deepening the GCN to propagate the features of the higher-order items to the target user. Thereafter, we introduced our two knowledge graph-



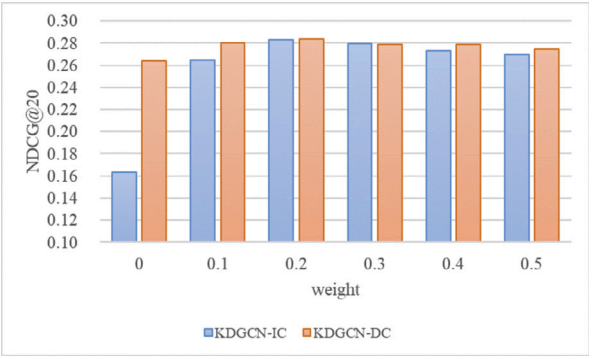
(a) Recall@20 on Book-Crossing



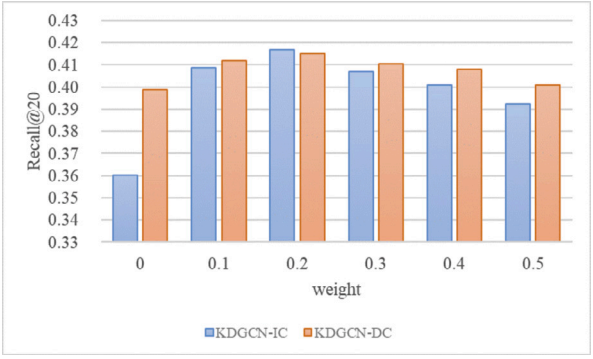
(b) NDCG@20 on Book-Crossing



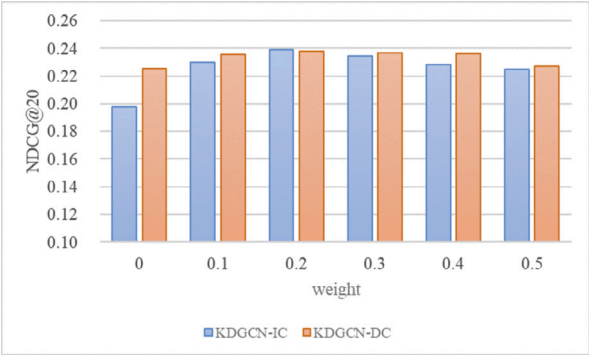
(c) Recall@20 on MovieLens-20M



(d) NDCG@20 on MovieLens-20M



(e) Recall@20 on Last-fm



(f) NDCG@20 on Last-fm

Fig. 8. Performance comparison *w.r.t.* different previous representation weight α_l.

aware deep GCN-based models to achieve high-order recommendation, named KDGCN-IC and KDGCN-DC. Our models utilize an item knowledge graph to rebuild the connectivity between items and the target user while building the connectivity between cold-start items and the target user, and reuse the features of previous layers to alleviate the over-smoothing issue in deepening the GCN. From the perspective of feature reuse, KDGCN-IC propagates the initial feature to each subsequent layer, and each layer in KDGCN-DC obtains features of the previous layers. We discussed extensive experiments we conducted on three widely-used benchmark datasets to demonstrate the potential of our models in cold-start scenarios and higher-order recommendations. The experimental results verified that our models maintained convincing performance in cold-start scenarios, and our models achieved remarkable improvement in high-order recommendation without almost losing the accuracy of low-order recommendation.

Our future work considers designing a relation-based attention mechanism to improve the feature aggregation process in knowledge graph-aware recommendations, that without introducing additional trainable parameters. In addition, believing that the deep layer might be redundant for the lower-order recommendation, a model that can adaptively select the depth during testing is a promising approach we are pursuing.

CRediT authorship contribution statement

Fei Wang: Conceptualization, Data curation, Formal analysis, Methodology, Validation, Visualization, Writing – original draft. **Zhi Zheng:** Validation, Writing – review & editing. **Yongjun Zhang:** Funding acquisition, Project administration, Supervision, Writing – review & editing. **Yansheng Li:** Funding acquisition, Project administration, Writing – review & editing. **Kun Yang:** Writing – review & editing. **Chenming Zhu:** Writing – review & editing.

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.

Acknowledgements

This work was supported by the National Natural Science Foundation of China [grant numbers 42030102, 41971284], the Fund for Innovative Research Groups of the Hubei Natural Science Foundation [grant number 2020CFA003], Major Special Project of Guizhou [grant number [2022]001], the Zhizhuo Research Fund on Spatial-Temporal Artificial Intelligence [grant number ZZJJ202210], and the Special Fund of Hubei Luojia Laboratory [grant number 220100032].

References

[1] R.v.d. Berg, T.N. Kipf, M. Welling, Graph convolutional matrix completion, arXiv preprint, arXiv:1706.02263, 2017.
[2] Y. Cao, X. Wang, X. He, Z. Hu, T.S. Chua, Unifying knowledge graph learning and recommendation: towards a better understanding of user preferences, in: The World Wide Web Conference, 2019, pp. 151–161.
[3] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, X. Sun, Measuring and relieving the over-smoothing problem for graph neural networks from the topological view, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2020, pp. 3438–3445.
[4] H. Chen, Y. Li, X. Sun, G. Xu, H. Yin, Temporal meta-path guided explainable recommendation, in: Proceedings of the 14th ACM International Conference on Web Search and Data Mining, 2021, pp. 1056–1064.
[5] 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, 2020, pp. 27–34.
[6] M. Chen, Z. Wei, Z. Huang, B. Ding, Y. Li, Simple and deep graph convolutional networks, in: International Conference on Machine Learning, PMLR, 2020, pp. 1725–1735.
[7] C. Gao, Y. Zheng, N. Li, Y. Li, Y. Qin, J. Piao, Y. Quan, J. Chang, D. Jin, X. He, et al., Graph neural networks for recommender systems: challenges, methods, and directions, arXiv preprint, arXiv:2109.12843, 2021.
[8] 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.
[9] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.
[10] Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, H. Xiong, Q. He, A survey on knowledge graph-based recommender systems, IEEE Trans. Knowl. Data Eng. (2020).
[11] 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.
[12] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, Adv. Neural Inf. Process. Syst. 30 (2017).
[13] 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.
[14] 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.
[15] G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4700–4708.
[16] D.P. Kingma, J.L. Ba, Adam: a method for stochastic optimization, in: International Conference on Learning Representations, ICLR, 2015.
[17] T.N. Kipf, M. Welling, Variational graph auto-encoders, arXiv preprint, arXiv:1611.07308, 2016.
[18] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: International Conference on Learning Representations, ICLR, 2017.
[19] J. Klicpera, A. Bojchevski, S. Günnemann, Predict then propagate: graph neural networks meet personalized pagerank, in: International Conference on Learning Representations, ICLR, 2019.
[20] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (2009) 30–37.
[21] G. Li, M. Muller, A. Thabet, B. Ghanem, Deepgcns: can gcns go as deep as cnns?, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 9267–9276.
[22] J. Li, Y. Rong, H. Cheng, H. Meng, W. Huang, J. Huang, Semi-supervised graph classification: a hierarchical graph perspective, in: The World Wide Web Conference, 2019, pp. 972–982.
[23] Q. Li, Z. Han, X.M. Wu, Deeper insights into graph convolutional networks for semi-supervised learning, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
[24] Y. Li, H. Chen, X. Sun, Z. Sun, L. Li, L. Cui, P.S. Yu, G. Xu, Hyperbolic hypergraphs for sequential recommendation, in: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, 2021, pp. 988–997.
[25] Y. Lin, Z. Liu, M. Sun, Y. Liu, X. Zhu, Learning entity and relation embeddings for knowledge graph completion, in: Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015.

[26] X. Liu, Y. Zhang, H. Zou, F. Wang, X. Cheng, W. Wu, X. Liu, Y. Li, Multi-source knowledge graph reasoning for ocean oil spill detection from satellite sar images, Int. J. Appl. Earth Obs. Geoinf. 116 (2023) 103153.
[27] Y. Liu, S. Yang, Y. Xu, C. Miao, M. Wu, J. Zhang, Contextualized graph attention network for recommendation with item knowledge graph, IEEE Trans. Knowl. Data Eng. (2021).
[28] K. Mao, J. Zhu, X. Xiao, B. Lu, Z. Wang, X. He, Ultragcn: ultra simplification of graph convolutional networks for recommendation, in: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, 2021, pp. 1253–1262.
[29] S. Rendle, Factorization machines, in: 2010 IEEE International Conference on Data Mining, IEEE, 2010, pp. 995–1000.
[30] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, Bpr: Bayesian personalized ranking from implicit feedback, arXiv preprint, arXiv:1205.2618, 2012.
[31] Y. Rong, W. Huang, T. Xu, J. Huang, Dropedge: towards deep graph convolutional networks on node classification, arXiv preprint, arXiv:1907.10903, 2019.
[32] P. Veličković, A. Casanova, P. Liò, G. Cucurull, A. Romero, Y. Bengio, Graph attention networks, in: International Conference on Learning Representations, ICLR, 2018.
[33] F. Wang, Y. Li, Y. Zhang, D. Wei, Klgcn: knowledge graph-aware light graph convolutional network for recommender systems, Expert Syst. Appl. (2022) 116513.
[34] G. Wang, R. Ying, J. Huang, J. Leskovec, Improving graph attention networks with large margin-based constraints, arXiv preprint, arXiv:1910.11945, 2019.
[35] 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.
[36] H. Wang, M. Zhao, X. Xie, W. Li, M. Guo, Knowledge graph convolutional networks for recommender systems, in: The World Wide Web Conference, 2019, pp. 3307–3313.
[37] 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.
[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] 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.
[40] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, K. Weinberger, Simplifying graph convolutional networks, in: International Conference on Machine Learning, PMLR, 2019, pp. 6861–6871.
[41] 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.
[42] S. Wu, F. Sun, W. Zhang, B. Cui, Graph neural networks in recommender systems: a survey, arXiv preprint, arXiv:2011.02260, 2020.
[43] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, T. Tan, Session-based recommendation with graph neural networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2019, pp. 346–353.
[44] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S.Y. Philip, A comprehensive survey on graph neural networks, IEEE Trans. Neural Netw. Learn. Syst. 32 (2020) 4–24.
[45] K. Xu, S. Jegelka, W. Hu, J. Leskovec, How powerful are graph neural networks?, in: International Conference on Learning Representations, ICLR, 2019.
[46] K. Xu, C. Li, Y. Tian, T. Sonobe, K.i. Kawarabayashi, S. Jegelka, Representation learning on graphs with jumping knowledge networks, in: International Conference on Machine Learning, PMLR, 2018, pp. 5453–5462.
[47] 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.
[48] 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.
[49] L. Zhao, L. Akoglu, Pairnorm: tackling oversmoothing in gnns, arXiv preprint, arXiv:1909.12223, 2019.
[50] K. Zhou, Y. Dong, W.S. Lee, B. Hooi, H. Xu, J. Feng, Effective training strategies for deep graph neural networks, arXiv e-prints, 2020.