Expert Systems With Applications 195 (2022) 116513

Contents lists available at ScienceDirect

Expert Systems With Applications

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

KLGCN: Knowledge graph-aware Light Graph Convolutional Network for recommender systems

Fei Wang, Yansheng Li [*], Yongjun Zhang [*], Dong Wei

School of Remote Sensing and Information Engineering, Wuhan University, Wuhan, Hubei, China

ARTICLE INFO

Keywords:
Recommender systems
Knowledge graph
Graph convolutional network
Collaborative filtering

ABSTRACT

Most popular recommender systems learn the embedding of users and items through capturing valuable information from user–item interactions or item knowledge graph (KG) with Graph Convolutional Network. However, only a few methods capture information from both source data, and they introduce many trainable parameters that increase training difficulty. In this work, we aim to aggregate information from both the user–item interaction graph and the item KG in a light and effective manner. We first experimentally verify the effectiveness of removing feature transformation and nonlinear activation in KG-aware recommendation, which has been proven to greatly reduce parameters while improving performance in the collaborative filtering-based recommendation. Then we propose a new *Knowledge graph-aware Light Graph Convolutional Network* (KLGCN), which can learn partial embeddings of users and items by aggregating features on the source graphs for recommendation and introduces no extra parameters. Extensive experiments on three public datasets demonstrate that KLGCN achieves substantial improvement over several state-of-the-art models and maintains satisfactory performance on cold-start scenarios.

1. Introduction

With the development of Internet technology, online applications have provided a multitude of content to users, such as books, movies and restaurants. The explosive growth of content has made applications notorious since it is difficult for users to pick out what they are really interested in. Recommender systems (RS) have been widely implemented to alleviate the negative effect of information overload by providing users a small set of items that meet their personalized interests (Covington, Adams, & Sargin, 2016; Wang et al., 2018; Ying et al., 2018).

The main task of RS is to predict whether a user will interact with an item, e.g. rate, click and view. Among different traditional recommendation strategies, collaborative filtering (CF)-based methods recommend items by considering historical interactions, which are shown as Fig. 1(a), to find the shared preference of users or the similarities of items, have achieved tremendous success in the past few years (Koren, Bell, & Volinsky, 2009; Wang, Wang, Zhao, Cao, & Guo, 2017). A main reason is the flourish of Graph Convolutional Networks (GCNs) which aggregate neighbor features by stacking multiple graph convolutional layers (GCLs) to capture the long-distance structural information and can be applied to graphs directly (Bruna, Zaremba, Szlam, & LeCun, 2013; Defferrard, Bresson, & Vandergheynst, 2016; Kipf & Welling, 2016a; Xu, Hu, Leskovec, & Jegelka, 2018). GCN and its variants have been proven to perform well in graph tasks, such as node classification (Abu-El-Haija, Kapoor, Perozzi, & Lee, 2020; Rong, Huang, Xu, & Huang, 2019), graph classification (Gao & Ji, 2019; Li et al., 2019) and link prediction (Kipf & Welling, 2016b). Applying GCN to user–item interaction graphs, the higher-order connections between nodes can be caught effectively (Berg, Kipf, & Welling, 2017; Chen, Wu, Hong, Zhang, & Wang, 2020). NGCF (Wang, He, Wang, Feng and Chua, 2019) explicitly injects the collaborative signal into the representation via following the standard propagation rule of GCN to model the high-order connectivity in user–item interaction graph and manifests promising improvement. LightGCN (He et al., 2020) simplifies NGCF via abandoning feature transformation and nonlinear activation, two operations standard in GCL but bring negative effect to CF, achieving state-of-the-art performance while reducing the complexity. That is to say, although GCN could benefit recommendation, the effect of its components are worth to be rethought.

In general, CF-based methods suffer from two challenges. One is the data sparsity, which is common in practical scenarios, such as movie

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding authors.
E-mail addresses: flyking@whu.edu.cn (F. Wang), yansheng.li@whu.edu.cn (Y. Li), zhangyj@whu.edu.cn (Y. Zhang), weidong@whu.edu.cn (D. Wei).

https://doi.org/10.1016/j.eswa.2022.116513
Received 4 July 2021; Received in revised form 3 January 2022; Accepted 4 January 2022
Available online 29 January 2022
0957-4174/© 2022 Elsevier Ltd. All rights reserved.

apps where the movie watched by a user is extremely few. The other is the cold-start, the lack of historical information makes it difficult for these methods to recommend new items. To solve these two limitations, researchers introduced side information to recommendation, such as contexts (Sun, Yuan, Xie, McDonald, & Zhang, 2017; Zhang, Yuan, Lian, Xie, & Ma, 2016), visual information (Zhang et al., 2016), user profiles/item attributes (Cao, Wang, He, Hu, & Chua, 2019; Wang et al., 2018), social networks (Ji, He, Xu, Liu, & Zhao, 2015) and so on.

In recent years, knowledge graphs (KGs) have attracted extensive attention in RS (Sang, Xu, Qian, & Wu, 2021; Wang, He, Cao, Liu and Chua, 2019; Wang et al., 2019; Wang, Zhao, Xie, Li and Guo, 2019) due to their abundant structural and semantic information between entities. A KG, as shown in Fig. 1(b), is a heterogeneous graph in which nodes represent entities (e.g., items or users, and their attributes or characteristics) and edges represent relations. The KG is helpful for recommendation with three aspects (Wang et al., 2018): (1) In the precision, the KG introduces fruitful semantic information for items, which is rewarding for deep exploration of user interests and precision improvement of item recommendation. (2) In the diversity, the KG provides a variety of relations which facilitate the diversification of recommended items. (3) In the explainability, the historical interacted items and the recommended items are connected through paths that consist of entities and relations, thus bringing explainability to recommendation as well as improving users' satisfaction. The proposal of several academic KGs (e.g., DBpedia, Freebase, NELL) has also profited the research (Guo et al., 2020).

Although KG-aware methods have achieved meaningful success, there are two general problems: (1) Focus on extracting information from KGs but ignore the collaborative signal contained in user–item historical interactions. However, these signals reflect the essential relation between users and items. (2) Introduce other complex models as aids (Lu & Altenbek, 2021; Wang, He, Cao et al., 2019), which triggers sharp increase in parameters and computational complexity.

Considering the aforementioned limitations of previous methods, we are fully convinced that it is of crucial importance to develop a method that can dig out useful information from both the interaction graph and the item KG in an intuitive and efficient approach. Inspired by the pattern of twin tower model (Cheng et al., 2016; Covington et al., 2016; He et al., 2017) which encodes various related features in parallel, we propose *Knowledge graph-aware Light Graph Convolutional Network* (KLGCN). KLGCN mainly contains two parallel GCL-based aggregation components which separately focus on the connectivity exploration of the two kinds of graphs.

- LGC. We inherit the aggregation layer of LightGCN — Light Graph Convolution (LGC) straightly to capture collaborative signals between users and items for its sightful performance in dealing with user–item interactions.
- LUCGC. We extend the aggregation layer of KGCN (Wang, Zhao et al., 2019) to Light User&Central nodes-specific Graph Convolution (LUCGC) which only keeps the most essential part of GCL – neighborhood aggregation – to aggregate features on item KGs. In the aggregation, a special designed attention mechanism is employed which simultaneously focus on the importance of relations to the specific users and the central nodes. In this way, not only users' personalized interests in relations are acquired, but also the semantic information is strengthened.

A simple weighted sum and inner product are used to the final recommendation. Empirically, we evaluate KLGCN on three real-world recommendation benchmark datasets: MovieLens-20M (movie), Book-crossing (book) and Last-FM (music). The experimental results show that comparing with state-of-the-art recommendation baselines, KLGCN achieves recall@20 gains of 12.45%, 57.05% and 9.83% on average in movie, book and music recommendation, respectively.

The contributions of this paper are summarized as follows:

- We experimentally verify that two essential designs in GCL – feature transformation and nonlinear activation – also contribute little to KG-aware recommendation.
- We develop a new model named KLGCN, which obtains information from the user–item historical interaction graph and the item KG simultaneously in a light and effective manner by discarding unnecessary components in GCL. Exploiting the information in both source data, the problems of cold-start when only using the user–item interactions and the insufficient of collaborative signals when only using the item KG can be effectively alleviated.
- We demonstrate the superiority of our model in top-$\mathcal{K}$ recommendation and cold-start scenarios via conducting extensive experiments and comprehensive analysis on three public datasets.

The remainder of this paper is organized as follows. Section 2 introduces the background and related work. In Section 3, we formalize the task. The details of the proposed method are described in Section 4. Section 5 reports the experimental results and analyzes the impact of hyperparameters and attention mechanism to the proposed model. In the last section, conclusions and future work are discussed.

2. Related work

We review and summarize existing work on CF-based recommendation and KG-aware recommendation that are most relevant to our work.

2.1. Collaborative filtering-based recommendation

CF is one of the most classical and prevalent techniques on recommendation which uses known preference of users to predict and recommend new preference for other users (Su & Khoshgoftaar, 2009). In general, CF techniques can be categorized as memory-based CF and model-based CF. The key idea of memory-based methods is K-nearest. These methods recommend same items to users with similar preference (Breese, Heckerman, & Kadie, 1998) or recommend similar items to a user (Linden, Smith, & York, 2003; Sarwar, Karypis, Konstan, & Riedl, 2001) via similarity measuring, like cosine similarity. However, they are unreliable when data is sparse.

The key idea of model-based approaches is representation gaining. An up-to-data approach is to embed users and items to trainable parameters and refine the parameters by rebuilding user–item historical interactions. Such as MF (Koren et al., 2009) maps ID feature of a user/an item to an embedding, FM (Rendle, 2010) and FFM (Juan, Zhuang, Chin, & Lin, 2016) use multi-features and consider the linear combination of them, but only low-level feature combinations can be obtained. To get the high-level feature combination, deep neural networks are jointly trained. Some distinguished models (Cheng et al., 2016; Covington et al., 2016; He et al., 2017; Zhou et al., 2018) share such a Embedding&MLP paradigm, and gain significant performance. However, the interaction of users and items are not explicit encoded because they are only used to define the loss function in these models, and therefore the collaborative signal is insufficient. A solution is to adopt GCN to capture the signal in the user–item interaction graph due to its strong performance on modeling graph structures. Pinsage (Ying et al., 2018) first introduces GCN into commercial recommendation and attains remarkable success. NGCF (Wang, He, Wang et al., 2019) follows the standard design of GCN to guide the embedding learning, LightGCN (He et al., 2020) further demonstrates that GCN could be applied in recommendation in a lighter and more effective way.

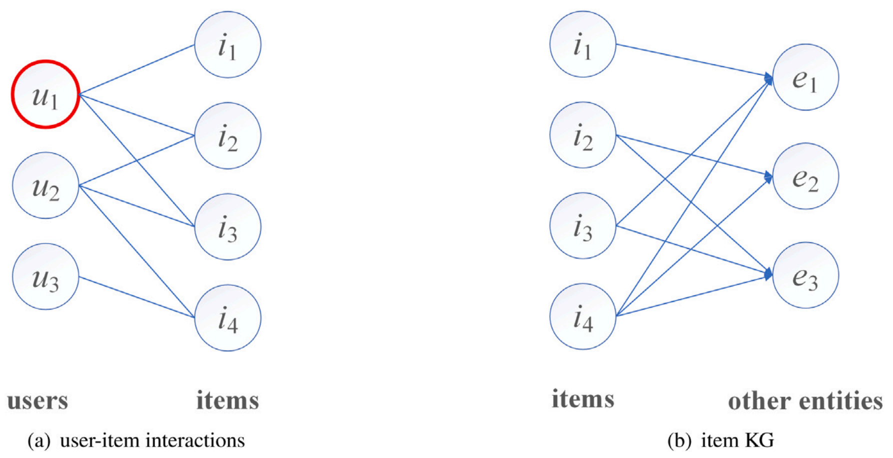


Fig. 1. An illustration of (a) user–item historical interactions and (b) item KG. A line in interactions indicates that the user interacted with the item. An arrow in KG indicates that there is a relation between the entities. The purpose is to provide recommendation for the target user u_1.

2.2. Knowledge graph-aware recommendation

KG is introduced to alleviate the cold-start problem and bring interpretability to recommendation. The best performing KG-aware methods can be classified into three groups: embedding-based, path-based and Graph Neural Network (GNN)-based. Embedding-based methods (Huang, Zhao, Dou, Wen, & Chang, 2018; Wang, Zhang, Xie and Guo, 2018; Wang et al., 2019; Zhang et al., 2016) encode the KG into low-dimension embeddings with knowledge graph embedding (KGE, e.g., TransE Bordes, Usunier, Garcia-Duran, Weston, & Yakhnenko, 2013, TransR Lin, Liu, Sun, Liu, & Zhu, 2015 and DistMult Yang, Yih, He, Gao, & Deng, 2014) and assign the learned embeddings to recommendation directly. However, these embeddings are might unsuitable for recommendation because they are unintuitive and inefficient in characterizing inter-item relations. Path-based methods (Lee, Kahng, & Lee, 2015; Palumbo, Monti, Rizzo, Troncy, & Baralis, 2020; Wang, Wang et al., 2019; Zhao, Yao, Li, Song, & Lee, 2017) explore high-order potential connections between entities in KGs via designing special meta paths/graphs to guide the recommendation. Although path-based methods explore the semantic information in KGs in an intuitive way and provide interpretability for recommendation, they require strong domain knowledge and the designed meta paths/graphs might just be suitable for special domains.

GNN-based methods encode the KG into low-dimension embeddings on the basic theory of signal processing on graphs. High-hop information can be captured through iterative aggregation, and the path formed by relations between nodes bring interpretability to recommendation. The emergency of GAT (Veličković et al., 2017) further facilitates the preference exploration in RS, because it makes it possible for GNN to distinguish the importance of different information. KGCN (Wang, Zhao et al., 2019) applies GCN to item KGs and use an attention mechanism to capture users' personalized preference on relations, KGNN-LS (Wang et al., 2019) provides a better inductive bias relies on label smoothness assumption to solve the overfitting that might occur in KGCN. KGCN and KGNN-LS show their potential on datasets, however, they focus on the connections between items but ignore the direct relationship between users and items. KGAT (Wang, He, Cao et al., 2019) applies GAT to the graph merged by the interaction graph and the item KG, and fine-tunes the gained embeddings with TransR. Although KGAT fully gains the collaborative signal in both source data and outperforms on datasets, the complexity increases due to the complex projection matrix of TransR.

Our method attempts to jointly aggregate the information in the user–item historical interaction graph and the item KG as light and effective as LightGCN does. To further highlight the uniqueness of our proposed model, we summarize and compare our model with some of the most related methods in Table 1 in terms of whether to introduce additional trainable weights, whether to gain information directly from the source data and the techniques adopted.

3. Problem formulation

The problem we focus on can be formulated as follows. In a typical recommendation scenario, we have a collection of historical interaction records of users to items. Let $\boldsymbol{U} = \{u_1, u_2, \ldots, u_N\}$ denotes a set of users and $\boldsymbol{I} = \{i_1, i_2, \ldots, i_M\}$ denotes a set of items. The user–item interaction matrix $\boldsymbol{Y} = \{(u,i)|u \in \boldsymbol{U}, i \in \boldsymbol{I}\}$ is defined according to the interaction records which represent the users' implicit feedback *w.r.t* items, and in which

$$y_{ui} = \begin{cases} 1, & \text{if interaction } (u,i) \text{ is observed,} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

A value 1 of y_{ui} indicates that user u implicitly interacts with item i, such as the behavior of watching, clicking and browsing, and 0 indicates that user u has not engaged with item i. Apart from the interaction matrix $\boldsymbol{Y}$, a corresponding KG $\boldsymbol{G} = \{(h,r,t)|h \in \mathcal{E}, r \in \boldsymbol{R}, t \in \mathcal{E}\}$ is also available which is constructed in the form of entity-relation-entity triples (h,r,t). For example, the triple (Gods Must Be Crazy, film.film.genre, comedy) states the fact that the genre of the film 'Gods Must Be Crazy' is comedy. $\mathcal{E} \in \mathbb{R}^O$ denotes a set of entities and $\boldsymbol{R} \in \mathbb{R}^Q$ denotes a set of relations.

In an item KG, an item $i \in \boldsymbol{I}$ matches with an entity $e \in \mathcal{E}$ in the KG. Taking 'Gods Must Be Crazy' as an instance, which is an item in the dataset MovieLens-20M as well as an entity in the corresponding movie KG. Therefore, the entity set $\mathcal{E}$ can be splitted into items $\boldsymbol{I}$ and non-items $\mathcal{E} \setminus \boldsymbol{I}$ (e.g., entities correspond to item characteristics). In detail, given a user–item interaction matrix $\boldsymbol{Y}$ and a KG $\boldsymbol{G}$, our task is to predict whether user u has potential interest in item i that the user has not interacted before. More specifically, we aim to predict the probability $\hat{y}_{ui}$ that user u will interact item i via learning a prediction function $\mathcal{F}(u, i|\boldsymbol{Y}, \boldsymbol{G}, \theta)$ where θ is the parameters.

4. Methodology

The architecture of KLGCN proposed in this paper is described in Fig. 2. There are three components of the architecture: (1) Embedding layer, which offers the initial embeddings of all users, entities and relations; (2) Aggregation layer, which is composed of LGC and LUCGC, and these two components refine embeddings by separately aggregating high-hop neighbor features with multiple layers on the user–item interaction graph and the item KG; and (3) Prediction layer, which generates the final embeddings through combining the partial refined

Table 1
Comparison of KLGCN and related methods. In the table, 'U-I' stands for user–item interaction graph, 'MLP' stands for multi-layer perceptron, 'Att.' stands for attention mechanism, 'AE' stands for autoencoder, 'MF' stands for matrix factorization.

Method	Weight matrix	Source data		Framework					
		U-I	KG	MLP	GCN	Att.	KGE	AE	MF
FM (Koren et al., 2009)	√								√
NeuMF (He et al., 2017)	√			√					√
NGCF (Wang, He, Wang et al., 2019)	√	√			√				
LightGCN (He et al., 2020)		√			√				
CKE (Zhang et al., 2016)	√		√				√	√	
RippleNet (Wang, Zhang, Wang et al., 2018)	√	√	√				√		
KGCN (Wang, Zhao et al., 2019)	√		√		√	√			
KGNN_LS (Wang, Zhang et al., 2019)	√		√		√	√			
KGAT (Wang, He, Cao et al., 2019)	√	√	√		√	√	√		
KLGCN		√	√		√	√			

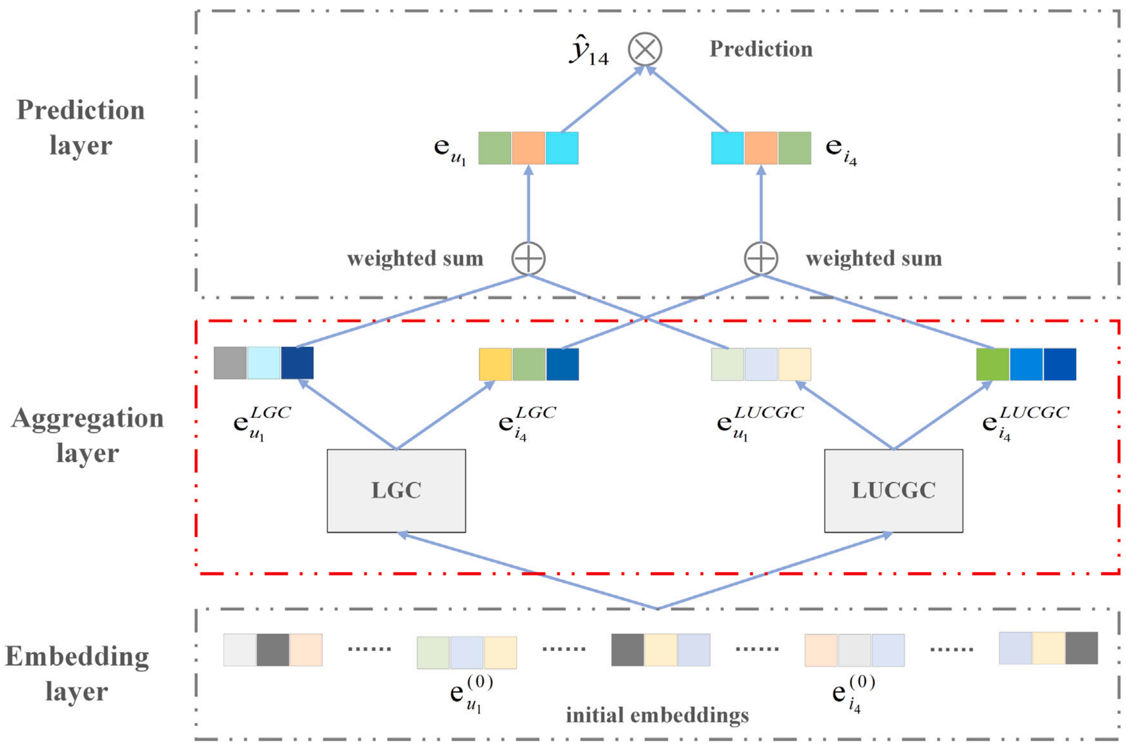


Fig. 2. Overall architecture of KLGCN. KLGCN is composed of embedding layer, aggregation layer and prediction layer. The initial embeddings are fed into the aggregation layer, and two partial embeddings of each user/item are obtained. A weighted sum of the partial embeddings are used to get the final embeddings. Recommend i_4 to u_1 is shown in the figure.

embeddings of users and items from the aggregation layer and outputs the score of a user–item pair which represents the probability of the user would interacts the item.

4.1. Embedding layer

To obtain the final appropriate representations, we assign each user (entity, relation) with an initial embedding vector $\mathbf{e}_u \in \mathbb{R}^d$ ($\mathbf{e}_e, \mathbf{e}_r \in \mathbb{R}^d$), where d is embedding size, which can be regarded as building a parameter matrix for embeddings looking up:

$$\mathbf{E} = \{\underbrace{\mathbf{e}_{u_1}, \mathbf{e}_{u_2}, \dots, \mathbf{e}_{u_N}}_{\text{users embeddings}}, \underbrace{\mathbf{e}_{i_1}, \mathbf{e}_{i_2}, \dots, \mathbf{e}_{i_M}}_{\text{items embeddings}}, \underbrace{\mathbf{e}_{e_{M+1}}, \mathbf{e}_{e_{M+2}}, \dots, \mathbf{e}_{e_O}}_{\text{entities}\backslash\text{items embeddings}}, \underbrace{\mathbf{e}_{r_1}, \mathbf{e}_{r_2}, \dots, \mathbf{e}_{r_Q}}_{\text{relations embeddings}}\}. \tag{2}$$

The initial embeddings can be optimized with our method in an end-to-end way. In contrast to traditional methods (He et al., 2017; Zhou et al., 2018) feed embeddings into interaction layers directly, our method propagates embeddings on the user–item interaction graph to capture the crucial collaborative signals to upgrade them satisfactory for CF, and we add auxiliary embeddings, which are gained by aggregating features on the item KG, as supplements to confront cold-start. In this way, the final embeddings contain abundant similarity information and semantic information.

4.2. Aggregation layer

Because of the sharing of signal propagation theory, we first introduce a standard GCL and then separately illustrate the two aggregation components — LGC and LUCGC, in which we focus on introducing the origin of LUCGC. Note that LGC contains the parts of embedding generation and embedding combination.

4.2.1. Standard LGC

The basic idea of GCN is to learn node representations by smoothing features over graphs. To achieve the target, it aggregates features of neighbor nodes and the target node itself as the new representation of the target node by stacking multiple convolutional layers. A multi-layer

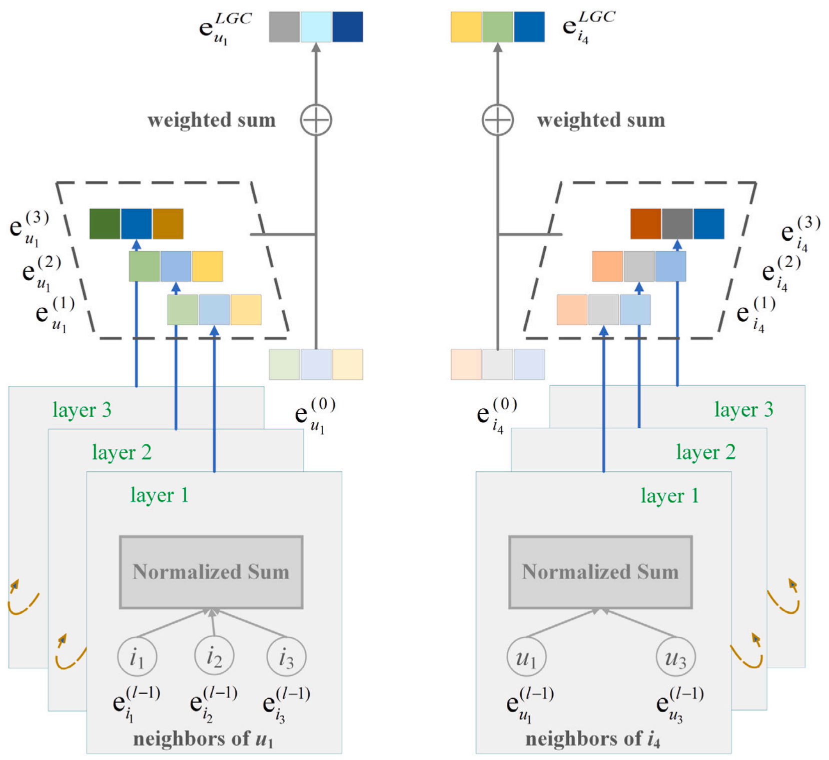


Fig. 3. An illustration of LGC architecture, which removes feature transformation and nonlinear activation in the aggregation, and a weighted sum of the embeddings at each layer are used to obtain the final embeddings. LGC works on user–item interaction graphs.

GCN follows the simple and well-behaved layer-wise propagation rule which operates directly on graphs (Kipf & Welling, 2016a):

$$\boldsymbol{H}^{(k+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}\boldsymbol{H}^{(k)}\boldsymbol{W}^{(k)}), \tag{3}$$

where $\tilde{A} = A + I$ denotes the graph adjacency matrix with added self-connections. In recommendation tasks, the user–item interaction graph is an undirected graph and the item KG is a directed graph. I is the identity matrix. $\tilde{D}$ is the diagonal degree matrix in which $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. $\boldsymbol{W}^{(k)}$ is a layer-wise trainable weight matrix. $\sigma(\cdot)$ is an nonlinear activation function as normal neural networks with, such as $ReLU(\cdot) = max(0, \cdot)$ (Glorot, Bordes, & Bengio, 2011). $\boldsymbol{H}^{(k)}$ is the feature of layer k, $\boldsymbol{H}^{(0)}$ is the initial feature $\boldsymbol{X}$ which can be set as an identity matrix I in featureless graph tasks, and can be set as initial user (item) embeddings $\{\mathbf{e}_{u_n}\}_{n=0}^{N}$ ($\{\mathbf{e}_{i_m}\}_{m=0}^{M}$) when it involves to RS. Following the rule, the lower-layer feature can be propagated to the higher-layer.

4.2.2. LGC

Most GCN-based works follow the standard form of GCL with feature transformation matrix $\boldsymbol{W}$ and nonlinear activation function $\sigma(\cdot)$. LightGCN (He et al., 2020) proves that these operations benefit to graph tasks such as node classification and graph classification due to their abundant semantic input features but might be cumbersome for CF-based recommendation since there is only ID feature in most cases.

The architecture of LGC is shown in Fig. 3, which abandons the feature transformation and nonlinear activation in the propagation process and just adopts a sum aggregator at each layer for the embedding generation of the target node, and is simply defined as:

$$\begin{aligned}\mathbf{e}_u^{(l+1)} &= \sum_{i\in\mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}}\mathbf{e}_i^{(l)},\\ \mathbf{e}_i^{(l+1)} &= \sum_{u\in\mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i||\mathcal{N}_u|}}\mathbf{e}_u^{(l)},\end{aligned} \tag{4}$$

where $\mathbf{e}_u^{(l)}$ and $\mathbf{e}_i^{(l)}$ denote the l_{th} layer embedding of user u and item i, respectively. $\mathcal{N}_u$ denotes the set of items in user–item interaction graph that have been interacted by user u and $|\mathcal{N}_u|$ denotes the number of the set. Similarly, $\mathcal{N}_i$ and $|\mathcal{N}_i|$ denote the set of users that have interacted with item i and its number, respectively. The symmetric normalization term $\frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}}$ is designed to avoid the explosion of the embedding scale, which plays a same role as the term $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ does in the standard GCL. It can be found that only the initial embeddings of all users $\{\mathbf{e}_{u_n}^{(0)}\}_{n=0}^{N}$ and items $\{\mathbf{e}_{i_m}^{(0)}\}_{m=0}^{M}$ are trainable parameters. After L layers feature aggregation, $L + 1$ embeddings are obtained which represent user u (item i) $\{\mathbf{e}_u^{(l)}\}_{l=0}^{L}$ ($\{\mathbf{e}_i^{(l)}\}_{l=0}^{L}$) from its initial embedding to the L-hop aggregated embedding. A weighted sum is used to combine the $L+1$ embeddings to get the final embedding of the user u (item i) and offset the influence of without self-connections:

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

where α_l denotes the weight of the l_{th} layer embedding in generating the final embedding which is experimentally set as a constant $1/(L+1)$. The embedding generation process of LGC is summarized in Algorithm 1.

4.2.3. User-specific Graph Convolution (UGC)

KGCN is a typical KG-aware recommendation model which aggregates features on the item KG and shows satisfactory performance over several datasets. The architecture of UGC is shown in Fig. 4(a). Generally speaking, users have different personalized interests in relations, e.g., a user might care more about 'genre' while someone else cares more about 'actor'. Therefore, the personalized interest of users is considered in the aggregation process via a special designed attention mechanism. A user-specific attention function $g : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is used

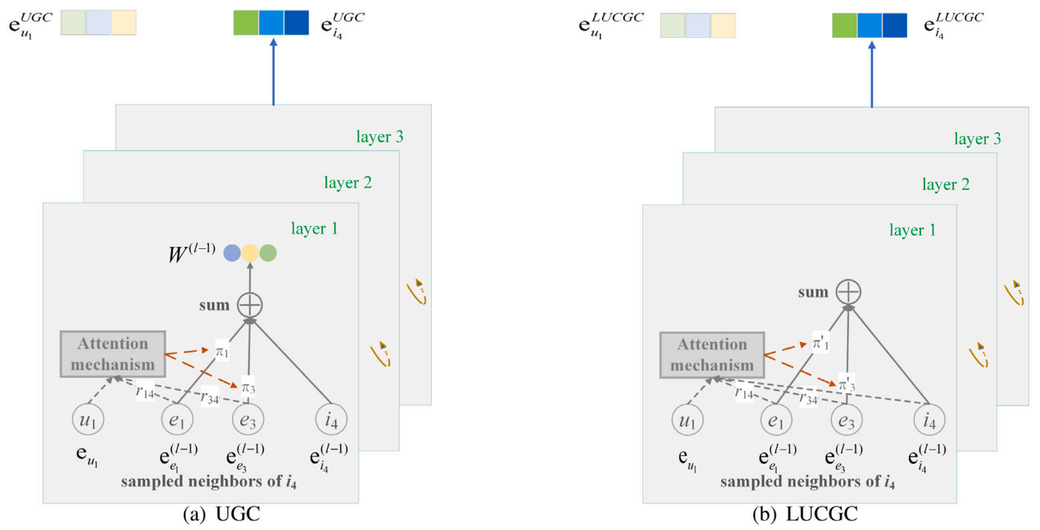


Fig. 4. An illustrate of (a) UGC architecture and (b) LUCGC architecture. UGC aggregates neighbor features with the design of standard GCL and a user-specific attention mechanism. LUCGC removes feature transformation and nonlinear activation on the basis of UGC and further designs a user&neighbor-specific attention mechanism. UGC and LUCGC work on item KGs.

Algorithm 1 Embedding generation process of LGC

```
Input: a user-item pair $(u, i)$; user-item interaction matrix $\boldsymbol{Y}$; initial embeddings of all users and items $\{\mathbf{e}_{u_n}\}_{n=0}^{N}$, $\{\mathbf{e}_{i_m}\}_{m=0}^{M}$; hyperparameters: layers $L$, layer weights $\{\alpha_l\}_{l=0}^{L}$;
Output: the user embedding $\mathbf{e}_u^{LGC}$; the item embedding $\mathbf{e}_i^{LGC}$;
1: $\mathcal{N}_u, \mathcal{N}_i \leftarrow \boldsymbol{Y}$    ▷ get neighbors of user $u$ and item $i$;
2: for $l = 1, 2, \cdots, L$ do
3:     $\mathbf{e}_u^{(l)} = \sum_{i\in\mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}}\mathbf{e}_i^{(l-1)}$;    ▷ propagate the $l$-hop neighbor features to user $u$;
4:     $\mathbf{e}_i^{(l)} = \sum_{u\in\mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i||\mathcal{N}_u|}}\mathbf{e}_u^{(l-1)}$;    ▷ propagate the $l$-hop neighbor features to item $i$;
5: $\mathbf{e}_u^{LGC} = \sum_{l=0}^{L}\alpha_l\mathbf{e}_u^{(l)}$;    ▷ generate the final LGC embedding of user $u$;
6: $\mathbf{e}_i^{LGC} = \sum_{l=0}^{L}\alpha_l\mathbf{e}_i^{(l)}$;    ▷ generate the final LGC embedding of item $i$;
7: return $\mathbf{e}_u^{LGC}, \mathbf{e}_i^{LGC}$
```

to calculate the care score of a user to a relation:

$$\pi^u_{r_{ei}} = g(u, r_{ei}), \tag{6}$$

where r_{ei} denotes the relation between entity e and item i, i.e., $e \in \mathcal{N}_i$ and $e \xrightarrow{r_{ei}} i$, $\mathcal{N}_i$ is the neighbor entities of item i in item KG. Considering that a KG in real-world is extremely large, where contains myriad entities and an entity may be linked up with thousands of entities, to keep the computational pattern fixed and efficient, UGC follows GraphSAGE (Hamilton, Ying, & Leskovec, 2017) to uniformly and randomly sample a fixed-size number K of neighbor nodes as the local neighborhood instead of using the full neighbors, which is similar to the receptive field of Convolution Neural Network in conceptual. In this way, a new neighbor set $\mathcal{S}_i$ is obtained where $|\mathcal{S}_i| \equiv K$, and duplicates are contained in the condition of $|\mathcal{N}_i| < K$.

UGC aggregates neighbor embeddings to generate an integrated representation $\mathbf{e}_{\mathcal{S}_i}$ as:

$$\mathbf{e}^{(h+1)}_{\mathcal{S}_i} = \sum_{e\in\mathcal{S}_i} \tilde{\pi}^u_{r_{ei}}\mathbf{e}^{(h)}_e, \tag{7}$$

where $\tilde{\pi}^u_{r_{ei}}$ is the normalized user-relation importance:

$$\tilde{\pi}^u_{r_{ei}} = \frac{\exp(\pi^u_{r_{ei}})}{\sum_{e'\in\mathcal{S}_i}\exp(\pi^u_{r_{e'i}})}. \tag{8}$$

To propagate the lower-layer embedding to the higher-layer, UGC follows the standard GCL design, applying feature transformation and nonlinear activation to feature generation.

$$\begin{aligned} \mathbf{e}^{(h+1)}_u &= \mathbf{e}^{(0)}_u, \\ \mathbf{e}^{(h+1)}_i &= \sigma(\boldsymbol{W}^{(h)}(\mathbf{e}^{(h)}_i + \mathbf{e}^{(h+1)}_{\mathcal{S}_i}) + b^{(h)}), \end{aligned} \tag{9}$$

where $b^{(h)}$ is the bias of the h_{th} layer. The transformation matrix and bias of each layer are shared. From Eq. (9) we can see that self-connection is added and the item embedding is updated in each iteration.

4.2.4. Empirical explorations on KGCN

To explore the influence of feature transformation and nonlinear activation on KG-aware recommendation, we conduct extensive ablation studies on KGCN. We use the codes[1] released by the authors of KGCN on model part, and use the codes[2] on data generation part for training acceleration. For fair comparison, we run experiments on same data split and evaluation metrics, each experiment is repeated five times, and the average performance is reported. The base model and its variants are as follows:

- KGCN, which keeps the feature transformation matrix $\boldsymbol{W}^{(h)}$ and nonlinear activation function $\sigma(\cdot)$.
- KGCN-f, which removes the feature transformation matrix but keeps the nonlinear activation function:

$$\mathbf{e}^{(h+1)}_i = \sigma(\mathbf{e}^{(h)}_i + \mathbf{e}^{(h+1)}_{\mathcal{S}_i}). \tag{10}$$

- KGCN-n, which removes the nonlinear activation function but keeps the feature transformation matrix:

$$\mathbf{e}^{(h+1)}_i = \boldsymbol{W}^{(h)}(\mathbf{e}^{(h)}_i + \mathbf{e}^{(h+1)}_{\mathcal{S}_i}) + b^{(h)}. \tag{11}$$

- KGCN-fn, which removes both the feature transformation matrix and nonlinear activation function:

$$\mathbf{e}^{(h+1)}_i = \mathbf{e}^{(h)}_i + \mathbf{e}^{(h+1)}_{\mathcal{S}_i}. \tag{12}$$

Note that only the generation manner of item representations has changed. We set each hyperparameter of KGCN and its three variants with same value, such as learning rate, embedding size, sampled

[1] https://github.com/hwwang55/KGCN.
[2] https://github.com/wubinzzu/NeuRec.

Table 2
Performance of KGCN and its three variants.

	MovieLens-20M		Last-fm	
	recall@20	ndcg@20	recall@20	ndcg@20
KGCN	0.3115	0.2485	0.3567	0.1964
KGCN-f	0.3325	**0.2676**	0.3574	0.2025
KGCN-n	0.3003	0.2371	0.3428	0.1862
KGCN-fn	**0.3472**	0.2666	**0.3651**	**0.2066**

neighbor number, regularization coefficient and so on, to ensure that the only experimental influencing factor is the component of GCL. We give an account of the results of the best hyperparameter settings on MovieLens-20M and Last-fm in Table 2, where the layer of UGC on the two datasets is set to 1 and the sampled neighbor size is 16 and 8 respectively. As we can see, when feature transformation is removed, model performs better (KGCN-f performs better than KGCN and KGCN-fn performs better than KGCN-n). However, the role of nonlinear activation is unclear, because when we remove it on the basis of KGCN, the model (i.e., KGCN-n) performs worse, but removing it on the basis of KGCN-f, the performance of the model (i.e., KGCN-fn) is improved (apart from ndcg@20 of MovieLens-20M drops lightly, recall@20 of both datasets improve imperatively and ndcg@20 of Last-fm improves lightly). We conclude these findings that:

- Removing feature transformation brings positive effect to KGCN, adding it leads to performance setbacks (compare KGCN with KGCN-f, KGCN-n with KGCN-fn).
- Removing nonlinear activation brings positive effect when feature transformation is disabled (compare KGCN-f with KGCN-fn), but it brings negative effect when feature transformation is enable (compare KGCN with KGCN-n).
- Removing feature transformation and nonlinear activation simultaneously brings positive effect to KGCN (compare KGCN with KGCN-fn).

4.2.5. LUCGC

The former experiments demonstrate that feature transformation brings only heavy complexity and play a limit role in KG-aware recommendation, and removing nonlinear activation will benefit the performance if feature transformation is removed. Therefore, we extend UGC to LUCGC, which is shown in Fig. 4(b). In LUCGC, we abandon the feature transformation and nonlinear activation simultaneously. After H-hop aggregation, the final embeddings can be represented as:

$$\mathbf{e}_u^{LUCGC} = \mathbf{e}_u^{(0)},\qquad \mathbf{e}_i^{LUCGC} = \mathbf{e}_i^{(H)} + \mathbf{e}_{S_i}^{(H+1)}. \tag{13}$$

In conventional GATs, the importance of neighbors to central nodes are mainly considered. However, the relationship between relation r and central node plays a nonnegligible important role in KGs, because the relation determines what role the neighbor plays in the circumstance of the central node. Taking triples (USA, hadPresident, Obama) and (Malia, isDaughter, Obama) as examples. Two types of relations connect two different neighbors with a same central node. Because the neighbor 'Malia' can uniquely determine the central node 'Obama' with the aid of the relation 'isDaughter', but 'hadPresident' cannot help 'USA' infer 'Obama' since 'USA' had other presidents, it can be considered that 'isDaughter' helps 'Malia' paly a more important role than 'USA' when the central node is 'Obama'. In other words, 'isDaughter' contributes more to 'Obama' than 'hadPresident'. Considering the importance of relations to central nodes, we further implement the attention function as:

$$\pi = \alpha\pi_{r_{ei}}^{u} + \beta\pi_{r_{ei}}^{i},\qquad \pi_{r_{ei}}^{u} = g(u, r_{ei}),\qquad \pi_{r_{ei}}^{i} = g(i, r_{ei}), \tag{14}$$

where α and β are the weights of care scores $\pi_{r_{ei}}^{u}$ and $\pi_{r_{ei}}^{i}$ in constituting the final attention score, respectively. Both of them can be treated as hyperparameters like learning rate, to be tuned manually, or trainable parameters like embedding vectors, to be trained automatically. We find that setting $\alpha = \beta = 1$ leads to good performance in our experiments, hence we leave them as constants to keep our model simple. The embedding generation process of LUCGC is summarized in Algorithm 2.

Algorithm 2 Embedding generation process of LUCGC

Input: a user-item pair (u, i); item KG $\boldsymbol{G}$; initial embeddings $\mathbf{E}$; hyperparameters: layers H, sampled neighbor size K;
Output: the user embedding $\mathbf{e}_u^{LUCGC}$; the item embedding $\mathbf{e}_i^{LUCGC}$;

```
1: $\{\mathcal{N}_i^h\}_{h=0}^{H} \leftarrow$ GetNeighbors$(i, \boldsymbol{G}, H, K)$;   ▷ generate $H$-hop neighbors of item $i$;
2: for $h = 1, 2, \cdots, H$ do
3:     for $e$ in $\mathcal{N}_i^h$ do
4:         $\mathbf{e}_{S_e}^{(h)} = \sum_{e' \in S_e} \tilde{\pi}_{r_{e'e}}^{u} \mathbf{e}_e^{(h-1)}$;   ▷ aggregate the $(H-h)$-hop neighbor features of item $i$;
5:         $\mathbf{e}_e^{(h)} = \mathbf{e}_e^{(h-1)} + \mathbf{e}_{S_e}^{(h)}$;   ▷ add self-collection;
6: $\mathbf{e}_u^{LUCGC} = \mathbf{e}_u^{(0)}$;   ▷ get the final LUCGC embedding of user $u$;
7: $\mathbf{e}_i^{LUCGC} = \mathbf{e}_i^{(H)}$;   ▷ get the final LUCGC embedding of item $i$;
8: return $\mathbf{e}_u^{LUCGC}, \mathbf{e}_i^{LUCGC}$
9:
10: function GetNeighbors$(i, \boldsymbol{G}, H, K)$
11:     $\mathcal{N}_i^H = i$;
12:     for $h = H-1, \cdots, 0$ do
13:         $\mathcal{N}_i^h = \emptyset$;
14:         for $e$ in $\mathcal{N}_i^{h+1}$ do
15:             $\mathcal{N}_e \leftarrow e, \boldsymbol{G}$;   ▷ get neighbors of entity $e$;
16:             $S_e \leftarrow K, \mathcal{N}_e$;   ▷ sample K neighbors from the entire neighbors;
17:             $\mathcal{N}_i^h = \mathcal{N}_i^h \cup S_e$;   ▷ update the $(H-h)$-hop neighbors of item $i$;
18:     return $\{\mathcal{N}_i^h\}_{h=0}^{H}$
```

4.3. Prediction layer

After feeding initial embeddings to the aggregation layer, two user embeddings $\mathbf{e}_u^{LGC}$ and $\mathbf{e}_u^{LUCGC}$, two item embeddings $\mathbf{e}_i^{LGC}$ and $\mathbf{e}_i^{LUCGC}$ are obtained. To get the final embeddings, a weighted sum is used:

$$\mathbf{e}_u = \omega_1 \mathbf{e}_u^{LGC} + \mu_1 \mathbf{e}_u^{LUCGC},\qquad \mathbf{e}_i = \omega_2 \mathbf{e}_i^{LGC} + \mu_2 \mathbf{e}_i^{LUCGC}, \tag{15}$$

where ω and μ are the importance of embeddings generated by LGC and LUCGC, separately. We manually set $\omega_i = \mu_i = 0.5, i \in \{1, 2\}$ to avoid complicating our proposed KLGCN. It can be seen that no additional trainable parameters are introduced to our model.

The model prediction is defined as feeding the generated representations of a user $\mathbf{e}_u$ and an item $\mathbf{e}_i$ to a function $f : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ to get the probability of user u has potential interest in item i. In our model, inner product is applied due to its simplicity:

$$\hat{y}_{ui} = \mathbf{e}_u^T \mathbf{e}_i. \tag{16}$$

4.4. Optimization

We employ *Bayesian Personalized Ranking* (BPR) loss (Rendle, Freudenthaler, Gantner, & Schmidt-Thieme, 2012) to train the model parameters which encourages users interest more on interacted items than their unobserved ones:

$$\mathcal{L} = -\sum_{(u,i,j)\in\mathcal{O}} \ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda\|\theta\|_2^2, \tag{17}$$

where $\mathcal{O} = \{(u,i,j)|(u,i) \in \mathbf{Y}, y_{ui} = 1; (u,j) \in \mathbf{Y}, y_{uj} = 0\}$, $\sigma(\cdot) = (1 + exp(-\cdot))^{-1}$ is the logistic sigmoid, λ is a balancing hyperparameter which controls the strength of L_2 regularization to prevent overfitting, θ is the trainable parameters which is equal to the initial embeddings $\mathbf{E}$.

To facilitate understanding, we give a summarization of the aforementioned algorithm in Algorithm 3.

Algorithm 3 The overall recommendation process of KLGCN

```
Input: user-item interaction matrix $\boldsymbol{Y}$; item KG $\boldsymbol{G}$; initial embeddings
    $\mathbf{E}$; hyperparameters: $L$, $H$, $K$, $\{\alpha_l\}_{l=0}^{L}$, $\omega_1$, $\omega_2$, $\mu_1$, $\mu_2$; probability
    calculation function $f(\cdot)$;
Output: prediction function $\mathcal{F}(u,i|\boldsymbol{Y},\boldsymbol{G},\theta)$;
1: while KLGCN not converge do
2:     for $(u,i)$ in $\boldsymbol{Y}$ do
3:         $\mathbf{e}_u^{LGC}, \mathbf{e}_i^{LGC} = LGC((u,i), \boldsymbol{Y}, \mathbf{E}, L, \{\alpha_l\}_{l=0}^{L})$; ▷ generate the LGC
    embeddings by Algorithm 1;
4:         $\mathbf{e}_u^{LUCGC}, \mathbf{e}_i^{LUCGC} = LUCGC((u,i), \boldsymbol{G}, \mathbf{E}, H, K)$; ▷ generate the
    LUCGC embeddings by Algorithm 2;
5:         $\mathbf{e}_u = \omega_1\mathbf{e}_u^{LGC} + \mu_1\mathbf{e}_u^{LUCGC}$;    ▷ generate the final KLGCN
    embedding of user $u$;
6:         $\mathbf{e}_i = \omega_2\mathbf{e}_i^{LGC} + \mu_2\mathbf{e}_i^{LUCGC}$;    ▷ generate the final KLGCN
    embedding of item $i$;
7:         Calculate predicted probability $\hat{y}_{ui} = f(\mathbf{e}_u, \mathbf{e}_i)$
8:         Update parameters, i.e., initial embeddings $\mathbf{E}$, by gradient
     descent
9: return $\mathcal{F}$
```

5. Experiments and discussion

In this section, we evaluate the proposed KLGCN and present its performance on three real-world recommendation scenarios: movie, book and music.

5.1. Evaluation datasets and metrics

To assess the effectiveness of our method, we conduct extensive experiments on the following three benchmark datasets for movie, book and music recommendation: MovieLens-20M, Book-Crossing and Last-FM. All these three datasets are publicly accessible and widely used in academic. And the performance of KLGCN on various scale KGs can be well evaluated due to their variety in terms of size and sparsity.

- MovieLens-20M is an extensively used benchmark dataset for movie recommendation which consists of almost 20 million explicit historical rating records (ranging from 1 to 5) on the MovieLens website.
- Book-Crossing consists of approximately 1 million explicit rating records (ranging from 0 to 10) of book items in the Book-Crossing Community.
- Last-FM consists of nearly 93 thousand musician listening information from a set of 2 thousand users from Last.fm online music system.

Table 3
Statistics of datasets.

	MovieLens-20M	Book-Crossing	Last-fm
Users	116 733	1248	1251
Items	16 953	14 965	3846
Interactions	6 613 568	39 755	16 669
Sparsity	0.9967	0.9979	0.9965
Entities	102 569	77 903	9366
Relations	32	25	60
KG triples	499 474	151 500	15 518

Since these three datasets are explicit feedbacks, we transform them into implicit feedbacks, where each record is marked with 1 indicating that the user has rated the item positively and 0 indicating that the user has low opinion on the item or has never interacted with it before. The threshold of positive rating is 4 for MovieLens-20M, while we set no threshold for the other two datasets. For each dataset, we use the 10-core setting to ensure the data quality, i.e., each user interacts with at least 10 items.

We also need the corresponding item KGs in addition to the user–item interaction records. We use the item KGs provided by KGCN which are constructed using Satori, a commercial KG built by Microsoft. The statistics of the three datasets after cleaning and corresponding KGs are summarized in Table 3.

For each dataset, we split the positive ratings into training set and test set as the ratio of 8:2, more specifically, we randomly sample 80% positive historical interactions of each user to constitute training set and leave the rest as the test set. For each positive interaction, we randomly select an item, the user has not rated positively before, to pair with the user as the negative sample. We adopt recall@$\mathcal{K}$ and ndcg@$\mathcal{K}$ to evaluate the capabilities of top-$\mathcal{K}$ recommendation and preference ranking of our model, all the non-positive items are candidates of a user, and $\mathcal{K} = 20$ by default. The hyperparameters are determined by optimizing the recall@20 on a test data. We apply the training strategies and calculation methods of evaluation metrics to all the comparative experiments to ensure the fairness. Note that all the methods share the acceleration codes.

5.2. Baselines

We compare KLGCN with supervised learning (FM), KG-aware (CKE, KGCN, KGAT), GCN-based (NGCF, LightGCN) methods over the datasets to demonstrate the superiority. The details are as follows:

- MF (Koren et al., 2009) is a benchmark model for recommendation. IDs of users and items are treated as the input features, and BPR loss is applied to optimize the parameters.
- CKE (Zhang et al., 2016) enhances the item representations with structural knowledges from item KGs on the basis of MF. These knowledges are embedded via TransR.
- KGCN (Wang, Zhao et al., 2019) applies GCN to discover the high-order structural and semantic information from item KGs, and cross-entropy loss is used to refine the representations.
- KGAT (Wang, He, Cao et al., 2019) trains recommendation and KGE alternately. All the structural and semantic information are gained from the graph merged by the user–item graph and the item KG.
- NGCF (Wang, He, Wang et al., 2019) explores the collaborative signal in the user–item interaction graph with standard GCL designs.
- LightGCN (He et al., 2020) is a state-of-the-art GCN-based model, which abandons the useless operates for recommendation in GCN – feature transformation and nonlinear activation, and only keeps the most essential component – neighborhood aggregation.

5.3. Hyperparameters settings

The settings of hyperparameters largely affect the results. The embedding size is fixed to 64 for all models on all datasets, except KGAT 32 on MovieLens-20M due to its high complexity. The trainable parameters are randomly initialized with a uniform distribution by the Xavier initialization methods (Glorot & Bengio, 2010) which keeps the scale of the gradients roughly the same in all layers to faster training convergence. The range of the uniform distribution is $[-\frac{\sqrt{6}}{\sqrt{n_j+n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j+n_{j+1}}}]$, in which n_j is the size of layer j. Mini-batch Adam (Kingma & Ba, 2014) optimizer is adopted to optimize parameters due to its effectiveness in preventing the training from trapping in local optima, where the batch size is set as 1024. We apply a grid search for the following parameters: The learning rate is searched in $\{0.01, 0.005, 0.001, 0.0005, 0.0001\}$, the L_2 regularization coefficient λ is tuned in the range from 10^{-1} to 10^{-9}, the layer size of GCN applied models (i.e., KGCN, KGAT, NGCF and LightGCN) is tuned amongst $\{1, 2, 3, 4, 5\}$, and the neighbor sampled size of KGCN is searched in $\{2, 4, 8, 16, 32\}$. For KGCN, we keep the hidden dimension of each layer fixed as the embedding size. As to other models with multi-standard GCLs (i.e., KGAT and NGCF), the hidden dimension of the first layer is set as the embedding size, and that of behind layers are defined as half of their previous layers in general. We consume node and message dropout techniques for KGAT and NGCF, and the ratios all are set as 0.1. Moreover, early stopping is employed in MovieLens-20M due to its big scale, i.e., premature stopping if recall@20 does not increase for 10 successive epochs on the test set.

5.4. Results

5.4.1. Performance comparison with baselines

The results of top-$\mathcal{K}$ recommendation and preference ranking are reported in Table 4. From the results we have the following observations:

- KGCN achieves poor performance in the three datasets (except recall@20 of MovieLens-20M is a little better than MF). The reason may be that KGCN gains the connections between items but ignores that of users and items, and excessive trainable parameters in KGCN leads to overfitting which is harmful to the generalization ability of the model, thus limiting the performance.
- The suboptimal performance of MF indicates that the combination of ID features and inner product are insufficient for capturing high-order and complex relations, more features and additional information are necessary. Compare amongst models that apply item KG as an auxiliary. In contrast to CKE underperforms than MF in most cases, KGAT consistently outperforms NGCF on MovieLens-20M and Last-fm. It demonstrates that using KGE to fine-tune the original representations is better than adding the representations learned by KGE as supplementary information to enrich the original representations, which further prove that the learned KG embeddings via KGE are might not suitable for recommendation.
- Compare with NGCF, KGCN performs dramatically worse in Book-Crossing and Last-fm, it verifies that the semantic information gained from item KG is relatively weak because they lack the connection between the user and the item, which is essential for recommendation. Compare LightGCN with NGCF, the performance of LightGCN achieves significant improvement on the all three datasets, which proves again that removing feature transformation and nonlinear activation is conducive to recommendation. KLGCN and LightGCN, two LGC-based models, achieve top-2 performance in most cases, a main reason might be that they have few parameters and low complexity, which is beneficial for preventing overfitting.

Table 4
Overall performance comparison with baseline methods.

	MovieLens-20M		Book-Crossing		Last-fm	
	recall@20	ndcg@20	recall@20	ndcg@20	recall@20	ndcg@20
MF	0.3066	0.2601	0.1235	0.0791	0.3845	0.2274
CKE	0.3115	0.2520	0.1029	0.0709	0.3791	0.2208
KGCN	0.3115	0.2485	0.0898	0.0637	0.3567	0.1964
KGAT	0.3379	0.2777	0.1049	0.0663	0.4288	0.2485
NGCF	0.2985	0.2503	0.1203	0.0815	0.4049	0.2351
LightGCN	0.3306	0.2909	0.1406	0.0987	0.4230	0.2577
KLGCN	**0.3548**	**0.2972**	**0.1748**	**0.1221**	**0.4333**	**0.2621**

- Compare with all baselines, KLGCN outperforms by a significant margin over all datasets in most cases. Particularly, KLGCN improves over the best baselines *w.r.t* recall@20 by 5.00%, 24.32% and 1.05% in MovieLens-20M, Book-Crossing and Last-fm, respectively. As to ndcg@20, it surpasses by 2.17%, 23.71% and 1.71% in the three datasets, respectively. By jointly aggregating information from historical interaction graph and item KG, and applying the information to recommendation directly, KLGCN captures the high-order connectivity as well as enhances the collaborative signal and semantic information via supplementing feature similarities of items to achieve better performance, while KGCN only utilizes KG, NGCF and LightGCN only capture signals from interaction graph, KGAT and CKE refine the final embedding through KGE. It demonstrates that the interaction graph and the item KG are beneficial to each other in recommendation.

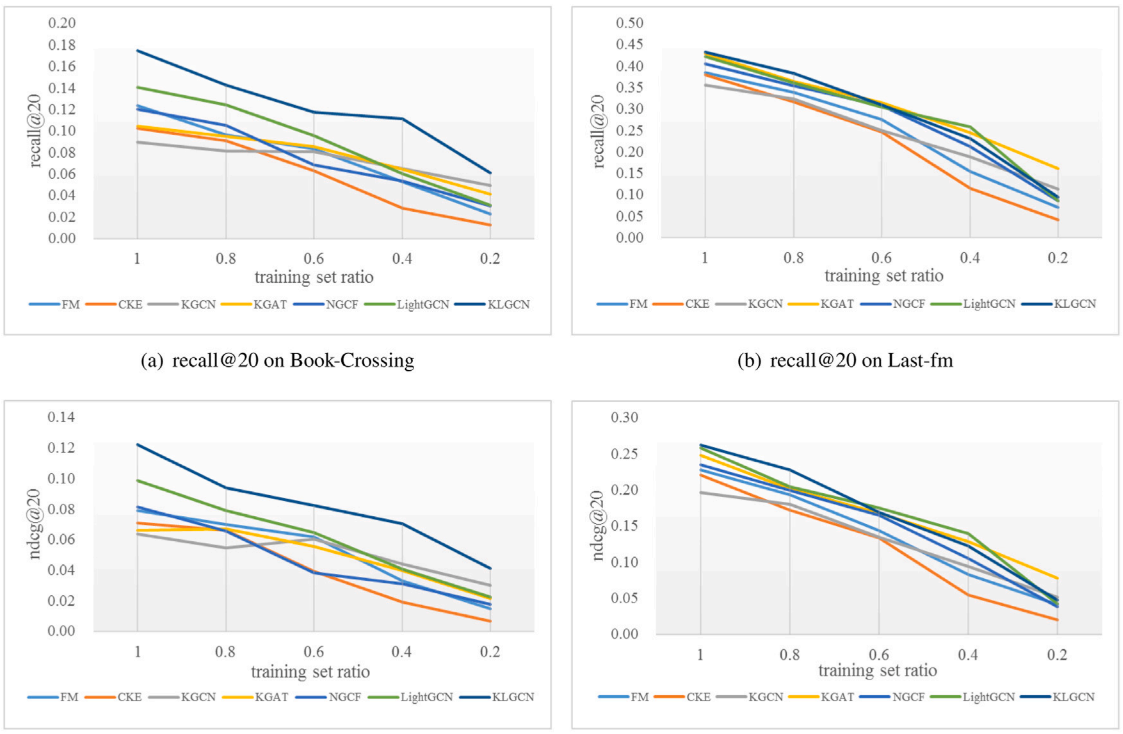
5.4.2. Performance comparison w.r.t cold-start scenarios

One of the main purposes of introducing the item KG to recommendation is to alleviate the limitation of cold-start. To explore the ability of KLGCN in confronting cold-start issues, we vary the ratio of training set to the original training set from 1.0 to 0.2 and keep the test set fixed. The results of Book-Crossing and Last-fm are shown in Fig. 5. Jointly analyzing the two subfigures, we have the following finds:

- The item KG can help resist the cold-start problem, especially in the case of extremely lack of data – when the ratio decreases to 0.2, the three best performing models are KGCN, KGAT and KLGCN, all of which aggregate features from the item KG – since the useful information about cold-start items, such as the connection about the cold-start items and the personalized interest of users in the cold-start items, can be injected into the final embedding, and therefore, the lack of information about these items can be made up for to a certain extent and the performance can be maintained.
- Our proposed model KLGCN is able to maintain superior performance in cold-start scenarios. In book recommendation, KLGCN shows the best performance in all sparsity. As to music recommendation, KLGCN shows the best performance when the ratio decreases to 0.8, and with the ratio decreases, it still maintains top-3 performance.

5.5. Ablation study of KLGCN

In GCN-based models, the layer of graph convolutional plays a vital role since it determines the depth of feature aggregation. It has been verified that 3 layers of LightGCN could lead to a satisfactory performance, so we keep the layer of LGC fixed and change the layer of LUCGC and explore its influence. We then study the sampled neighbor size of LUCGC which determines the width of feature aggregation. Thirdly, we investigate the effect of the attention mechanism of LUCGC. Finally, we simply explore the sensitivity *w.r.t* the weights used to generate the final embedding, i.e., ω and μ.



(a) recall@20 on Book-Crossing

(b) recall@20 on Last-fm

(c) ndcg@20 on Book-Crossing

(d) ndcg@20 on Last-fm

Fig. 5. Performance comparison *w.r.t* different ratio of the training set on Book-Crossing and Last-fm. Training set ratio indicates the ratio to the original training set.

5.5.1. *Impact of LUCGC layer number*

To explore how the layer number influences the performance, we vary it from 1 to 5 while keep other hyperparameters fixed. The results are summarized in Table 5, wherein LUCGC-li indicates LUCGC with i layers, the mark — indicates out-of-memory and hence without result. Analyzing Table 5 we have the following observations:

- The best performance can be achieved in a shallow layer – the optimal layers of the three datasets are 1, 4 and 1, respectively – since as the layer deepens, the expansion of neighbor nodes is inevitable, and there will be over-smoothing issues, i.e., the representation of nodes tends to be similar, which will cause the network to be unable to identity the item that users are interested in.
- Deeper layer means more positive information. Below the optimal layers, as the number of layers deeps from 1 to 4, recall@20 on Book-Crossing separately increases 2.23%, 10.34% and 11.41%.
- Deeper layer also means more negative information. Over the optimal layers, as the number of layers deeps from 1 to 3, recall@20 on MovieLens-20M separately decreases 6.54% and 0.27%, and on Last-fm, it drops 14.03% and 6.85%, respectively. A hypothesis can be proposed, that is with the layer deepens, the noise will gradually tend to be saturated, in other words, the performance will be stable.

Table 5
Performance of LUCGC with different embedding layers.

	MovieLens-20M		Book-Crossing		Last-fm	
	recall@20	ndcg@20	recall@20	ndcg@20	recall@20	ndcg@20
LUCGC-l1	**0.3548**	**0.2972**	0.1391	0.1020	**0.4333**	**0.2621**
LUCGC-l2	0.3316	0.2765	0.1422	0.1053	0.3725	0.2286
LUCGC-l3	0.3307	0.2794	0.1569	0.1146	0.3470	0.2184
LUCGC-l4	–	–	**0.1748**	**0.1221**	–	–
LUCGC-l5	–	–	0.1429	0.0955	–	–

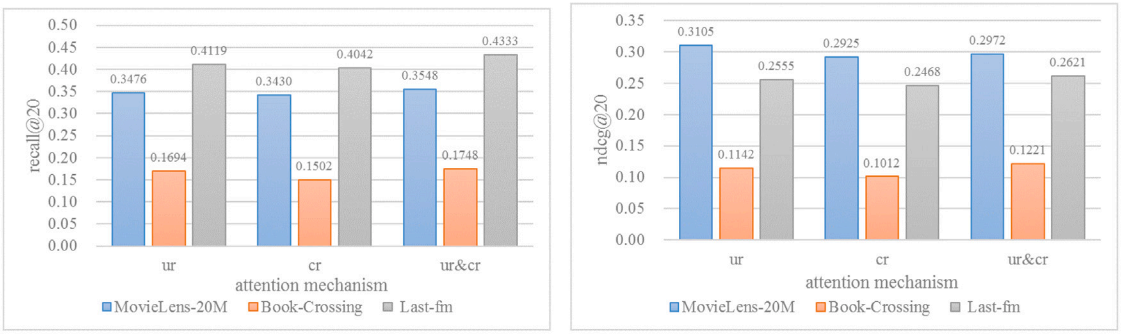
5.5.2. *Impact of LUCGC sampled neighbor number*

To investigate how the neighbor size affects the performance, we search it in the range of $\{2, 4, 8, 16, 32\}$. Table 6 shows the result of changing neighbor size in where LUCGC-n2 indicates that sampling 2 neighbors for each node from its neighbor set, and similar for other symbols. We can find that

- Aggregating features of the entire neighbors might be suboptimal because the performance drops as the neighbor size exceeds a threshold. The threshold can be set as 8 since the best performance is generally achieved when neighbor size below 8.
- Below the threshold, as the neighbor size increases, the performance of KLGCN improves significantly. For example, the neighbor number increases from 4 to 8, recall@20 on the MovieLens-20M and Last-fm increase by 4.17% and 9.17%, respectively. Because the important local information of partial nodes will be lost when the neighbor field is too small, and a larger neighbor filed will bring more positive information to the feature aggregation.
- As the threshold is exceeded, the performance of KLGCN drops quickly with the growing of neighbor size. Recall@20 of MovieLens-20M and Last-fm respectively drop by 5.48% and 7.47% when increasing neighbor number from 16 to 32, which signifies that more noises are aggregated and the negative impacts are gradually exceed the positive impacts.

5.5.3. *Impact of LUCGC attention mechanism*

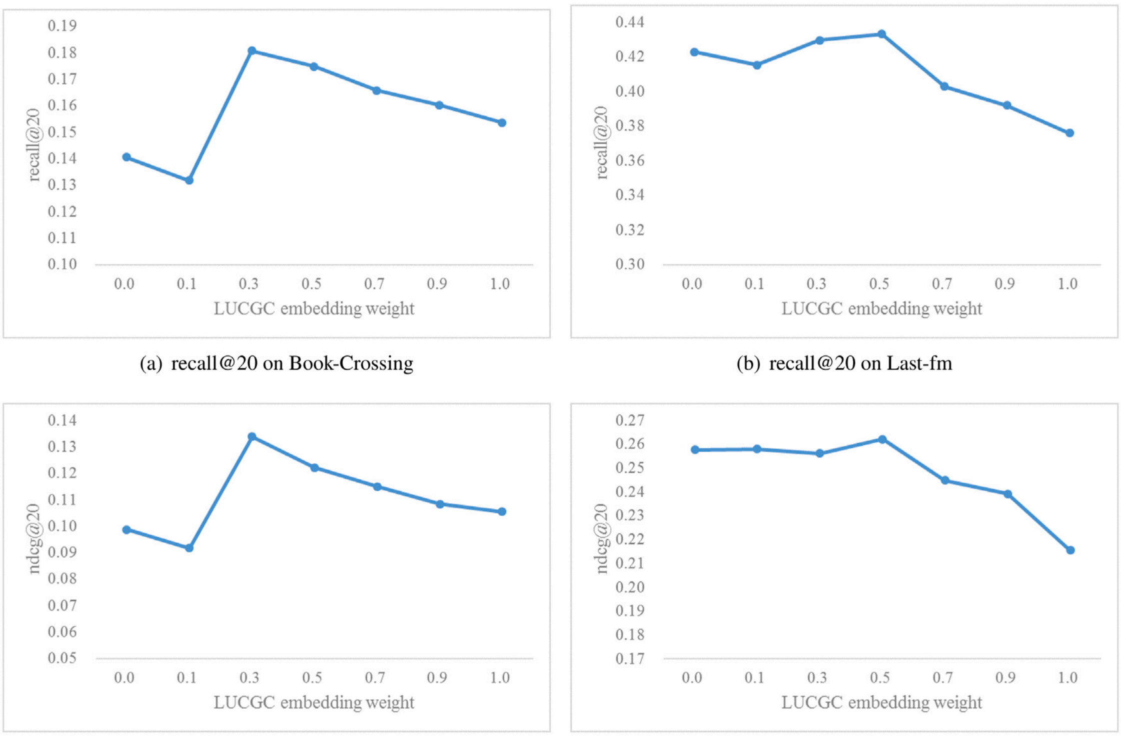
Neighbors contribute different to the central node, the attention mechanism could affect the model performance significantly. To explore its impact, we define various attention function by changing the values of α and β. ur indicates that $\alpha = 1, \beta = 0$, i.e., we only consider the personalized interests in relations. cr indicates that $\alpha = 0, \beta = 1$ which means that neighbor roles are considered only. $ur\&cr$ is the base attention mechanism in which $\alpha = \beta = 1$. The results are shown in Fig. 6, from which there are several observations we can find:



(a) recall@20 on the three datasets

(b) ndcg@20 on the three datasets

Fig. 6. Performance comparison $w.r.t$ different attention mechanism on MovieLens-20M, Book-Crossing and Last-fm.



(a) recall@20 on Book-Crossing

(b) recall@20 on Last-fm

(c) ndcg@20 on Book-Crossing

(d) ndcg@20 on Last-fm

Fig. 7. Performance $w.r.t$ different weight of the LUCGC embedding μ on Book-Crossing and Last-fm.

Table 6
Performance of KLGCN with different LUCGC sampled neighbors.

	MovieLens-20M		Book-Crossing		Last-fm	
	recall@20	ndcg@20	recall@20	ndcg@20	recall@20	ndcg@20
LUCGC-n2	0.3317	0.2847	0.1748	**0.1221**	0.3801	0.2328
LUCGC-n4	0.3406	0.2858	**0.1753**	0.1190	0.3970	0.2326
LUCGC-n8	**0.3548**	**0.2972**	–	–	**0.4333**	**0.2624**
LUCGC-n16	0.3501	0.2898	–	–	0.4043	0.2465
LUCGC-n32	0.3309	0.2640	–	–	0.3741	0.2313

- Except ndcg@20 on MovieLens-20M, our model performs best with *ur*&*cr* attention mechanism, which illustrates the importance of both the personalized interests (*ur*&*cr* performs better than *cr*) and neighbor roles (*ur*&*cr* outperforms *ur*).
- In all cases, our model with *ur* attention mechanism achieves consistent improvement in contrast to with *cr* attention mechanism, it verifies that personalized interests play an irreplaceable role in recommendation which are more important than neighbor roles.

5.5.4. Impact of embedding generation weights

The weights ω and μ determine the contribution of the partial embedding to the final embedding, thereby affecting the quality of the final embedding. We assume $\omega_1 = \omega_2$, $\mu_1 = \mu_2$ and $\omega_i + \mu_i = 1, i \in \{1, 2\}$ to simplify the sensitivity exploration of these weights. In-depth weight exploration for different datasets is encouraged. Performance change is shown in Fig. 7, from which we can find that:

- As the weight of LUCGC embedding increases, the performance shows a significant trend from falling to raising to falling, indicating that the final embedding generation is weight sensitive. Since the best performance is achieved in the process rather than when $\mu = 0$ (i.e., only aggregate information on user–item interactions) or $\mu = 1$ (i.e., only aggregate information on item KGs), it is necessary to combine the information from the two kinds of source data and an appropriate coupling is needed.
- The optimal value of the weight of LUCGC embedding for Book-Crossing and Last-fm is 0.3 and 0.5, respectively. It indicates that the collaborative signal from user–item interaction is more

important than the semantic information from item KG for recommendation when combining the two kinds of information, because the collaborative signal could better reflect the essential relationship between the user and the item, and it might be not enough to only reflect the user information in the attention mechanism or the loss function.

6. Conclusions and future work

In this paper, we explore the high-order structural and semantic information in both the user–item interactions and the item KG for recommendation in a light and effective manner. We propose a new model named KLGCN which separately aggregates features from the two kinds of source data with an end-to-end fashion. The core idea behind the aggregation layer of KLGCN is removing the feature transformation and nonlinear activation, which are standard in GCL but bring negative effect to recommendation as well as highly increase the trainable parameters, but just adopting a sum aggregator for embedding generation. In the prediction layer, a weighted sum is used to combine the output embeddings of the aggregation layer to fully merge the collaborative signal in the interaction graph and the semantic information in the item KG. Extensive experiments demonstrate that KLGCN significantly outperforms state-of-the-art methods in MovieLens-20M, Book-Crossing and Last-fm datasets.

We point out four directions for future works: (1) Considering that aggregating features of the entire neighbors might be unsuitable for large graphs and suboptimal for recommendation, the strategy of sampling neighbors can be applied to both components of the aggregation layer. Furthermore, exploring a reasonable sampler for neighbor sampling rather than random sampling is an important direction. (2) Considering neighbors contribute different to the central node, attention mechanism can be also utilized in the feature aggregation process on the user–item interaction graph. (3) Experimental results have proved that comparing to the model which only uses interaction graphs, the model uses item KGs as auxiliary information can improve the performance, so introducing more types of supplementary data, like visual data, for recommendation is a promising direction. (4) Considering the problems of time-consuming and resource consumption in BP-based neural networks (Wang & Cao, 2018), introducing non-iterative training methods (Cao, Hu, Gao, Wang, & Ming, 2020; Cao, Wang, Ming, & Gao, 2018) is a direction worth trying.

CRediT authorship contribution statement

Fei Wang: Conceptualization, Methodology, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Visualization. **Yansheng Li:** Validation, Writing – review & editing, Project administration, Funding acquisition. **Yongjun Zhang:** Writing – review & editing, Supervision, Project administration, Funding acquisition. **Dong Wei:** Writing – review & editing, Project administration.

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.

Acknowledgments

This work is partly supported by the National Key Research and Development Program of China under grant 2018YFB0505003; the National Natural Science Foundation of China under grant 42030102 and 41971284.

References

Abu-El-Haija, S., Kapoor, A., Perozzi, B., & Lee, J. (2020). N-gcn: Multi-scale graph convolution for semi-supervised node classification. In *Uncertainty in artificial intelligence* (pp. 841–851). PMLR.

Berg, R. v. d., Kipf, T. N., & Welling, M. (2017). Graph convolutional matrix completion. arXiv preprint arXiv:1706.02263.

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., & Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In *Neural information processing systems (NIPS)* (pp. 1–9).

Breese, J. S., Heckerman, D., & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the fourteenth conference on uncertainty in artificial intelligence* (pp. 43–52).

Bruna, J., Zaremba, W., Szlam, A., & LeCun, Y. (2013). Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203.

Cao, W., Hu, L., Gao, J., Wang, X., & Ming, Z. (2020). A study on the relationship between the rank of input data and the performance of random weight neural network. *Neural Computing and Applications*, 1–12.

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

Cao, W., Wang, X., Ming, Z., & Gao, J. (2018). A review on neural networks with random weights. *Neurocomputing, 275*, 278–287.

Chen, L., Wu, L., Hong, R., Zhang, K., & Wang, M. (2020). Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 27–34).

Cheng, H.-T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., et al. (2016). Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems* (pp. 7–10).

Covington, P., Adams, J., & Sargin, E. (2016). Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems* (pp. 191–198).

Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. arXiv preprint arXiv:1606.09375.

Gao, H., & Ji, S. (2019). Graph u-nets. In *International conference on machine learning* (pp. 2083–2092). PMLR.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 249–256). JMLR Workshop and Conference Proceedings.

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics* (pp. 315–323). JMLR Workshop and Conference Proceedings.

Guo, Q., Zhuang, F., Qin, C., Zhu, H., Xie, X., Xiong, H., et al. (2020). A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*.

Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. arXiv preprint arXiv:1706.02216.

He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., & Wang, M. (2020). 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* (pp. 639–648).

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T.-S. (2017). Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web* (pp. 173–182).

Huang, J., Zhao, W. X., Dou, H., Wen, J.-R., & Chang, E. Y. (2018). Improving sequential recommendation with knowledge-enhanced memory networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval* (pp. 505–514).

Ji, G., He, S., Xu, L., Liu, K., & Zhao, J. (2015). Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (Volume 1: Long papers)* (pp. 687–696).

Juan, Y., Zhuang, Y., Chin, W.-S., & Lin, C.-J. (2016). Field-aware factorization machines for CTR prediction. In *Proceedings of the 10th ACM conference on recommender systems* (pp. 43–50).

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kipf, T. N., & Welling, M. (2016a). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

Kipf, T. N., & Welling, M. (2016b). Variational graph auto-encoders. arXiv preprint arXiv:1611.07308.

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

Lee, S., Kahng, M., & Lee, S.-g. (2015). Constructing compact and effective graphs for recommender systems via node and edge aggregations. *Expert Systems with Applications*, *42*(7), 3396–3409.

Li, J., Rong, Y., Cheng, H., Meng, H., Huang, W., & Huang, J. (2019). Semi-supervised graph classification: A hierarchical graph perspective. In *The world wide web conference* (pp. 972–982).

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

Linden, G., Smith, B., & York, J. (2003). Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, *7*(1), 76–80.

Lu, W., & Altenbek, G. (2021). A recommendation algorithm based on fine-grained feature analysis. *Expert Systems with Applications*, *163*, Article 113759.

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

Rendle, S. (2010). Factorization machines. In *2010 IEEE international conference on data mining* (pp. 995–1000). IEEE.

Rendle, S., Freudenthaler, C., Gantner, Z., & Schmidt-Thieme, L. (2012). BPR: Bayesian personalized ranking from implicit feedback. arXiv preprint arXiv:1205.2618.

Rong, Y., Huang, W., Xu, T., & Huang, J. (2019). Dropedge: Towards deep graph convolutional networks on node classification. arXiv preprint arXiv:1907.10903.

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

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on world wide web* (pp. 285–295).

Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, *2009*.

Sun, Y., Yuan, N. J., Xie, X., McDonald, K., & Zhang, R. (2017). Collaborative intent prediction with real-time contextual data. *ACM Transactions on Information Systems (TOIS)*, *35*(4), 1–33.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. arXiv preprint arXiv:1710.10903.

Wang, X., & Cao, W. (2018). Non-iterative approaches in training feed-forward neural networks and their applications. *Soft Computing*, *22*, 3473–3476.

Wang, X., He, X., Cao, Y., Liu, M., & Chua, T.-S. (2019). Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 950–958).

Wang, X., He, X., Wang, M., Feng, F., & Chua, T.-S. (2019). Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval* (pp. 165–174).

Wang, J., Huang, P., Zhao, H., Zhang, Z., Zhao, B., & Lee, D. L. (2018). Billion-scale commodity embedding for e-commerce recommendation in alibaba. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 839–848).

Wang, X., Wang, D., Xu, C., He, X., Cao, Y., & Chua, T.-S. (2019). Explainable reasoning over knowledge graphs for recommendation. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 5329–5336).

Wang, H., Wang, J., Zhao, M., Cao, J., & Guo, M. (2017). Joint topic-semantic-aware social recommendation for online voting. In *Proceedings of the 2017 ACM on conference on information and knowledge management* (pp. 347–356).

Wang, H., Zhang, F., Hou, M., Xie, X., Guo, M., & Liu, Q. (2018). Shine: Signed heterogeneous information network embedding for sentiment link prediction. In *Proceedings of the eleventh ACM international conference on web search and data mining* (pp. 592–600).

Wang, H., Zhang, F., Wang, J., Zhao, M., Li, W., Xie, X., et al. (2018). Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *Proceedings of the 27th ACM international conference on information and knowledge management* (pp. 417–426).

Wang, H., Zhang, F., Xie, X., & Guo, M. (2018). DKN: Deep knowledge-aware network for news recommendation. In *Proceedings of the 2018 world wide web conference* (pp. 1835–1844).

Wang, H., Zhang, F., Zhang, M., Leskovec, J., Zhao, M., Li, W., et al. (2019). 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* (pp. 968–977).

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

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

Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2018). How powerful are graph neural networks? arXiv preprint arXiv:1810.00826.

Yang, B., Yih, W.-t., He, X., Gao, J., & Deng, L. (2014). Embedding entities and relations for learning and inference in knowledge bases. arXiv preprint arXiv:1412.6575.

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., & Leskovec, J. (2018). Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 974–983).

Zhang, F., Yuan, N. J., Lian, D., Xie, X., & Ma, W.-Y. (2016). Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 353–362).

Zhao, H., Yao, Q., Li, J., Song, Y., & Lee, D. L. (2017). Meta-graph based recommendation fusion over heterogeneous information networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 635–644).

Zhou, G., Zhu, X., Song, C., Fan, Y., Zhu, H., Ma, X., et al. (2018). Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 1059–1068).