



BIKAGCN: Knowledge-Aware Recommendations Under Bi-layer Graph Convolutional Networks

Guoshu Li¹ · Li Yang¹ · Sichang Bai¹ · Xinyu Song¹ · Yijun Ren¹ · Shanqiang Liu¹

Accepted: 31 October 2023 / Published online: 7 February 2024
© The Author(s) 2024

Abstract

Recommender systems are a popular solution for the problem of information overload, offering personalized recommendations to users. Recent years, research has aimed to enhance recommender systems by employing knowledge graphs in conjunction with Graph convolutional network (GCN) to extract user and item features. Although GCN possess a great potential, they are still far from reaching their full capability in recommender systems. This paper introduces a novel approach—knowledge-aware recommendations under bi-layer graph convolutional networks (BIKAGCN) that combines attention and bi-layer GCNs to improve performance. The first layer of the BIKAGCN model trains embedding representations of users and items based on user-item interaction graphs. The second layer introduces a novel knowledge-aware layer of attention and graph convolutional network (KAGCN) layer that leverages both the first layer's user-item embeddings and item knowledge graph embeddings. Experimental results on three publicly available datasets (MovieLens-20M, Last-FM, and Book-Crossing) demonstrate that BIKAGCN leads to significant performance improvements in recall@20 metric (14.41%, 8.86%, and 20.90%, respectively) compared to currently available state-of-the-art approaches. Moreover, the model maintains satisfactory performance in cold-start cases. The research provides some guidance for the direction of subsequent research on recommender systems.

Keywords Knowledge graph · Recommender systems · Graph convolutional network · Collaborative filtering

1 Introduction

With the geometric growth of multimedia information, driven by the growth of Internet technology, has surpassed humans' capacity to effectively process and utilize information

✉ Li Yang
xnsy_ly@swpu.edu.cn

Guoshu Li
18228037494lgs@gmail.com

¹ School of Computer Science and Software Engineering, Southwest Petroleum University, Xindu Avenue, Chengdu 610500, Sichuan, China

within a brief period. To address this challenge, Recommender systems (RS) have emerged as a valuable tool to assist users in obtaining the relevant information they need. RS have been applied in various fields, such as news [1], movies [2], and products [3].

The core of a recommender system is to estimate the probability that a user is interested in an item based on his or her historical interaction behavior with the item. Most of the existing recommendation techniques use embedding techniques to simulate the interaction between user and item, such as using inner product [4] and neural networks [5]. Collaborative filtering (CF) uses past user-item interactions to make recommendations and has received significant attention in recent years [4, 6–11]. However, collaborative filtering based recommendation methods suffer from data sparsity and cold-start.

Recently, knowledge graph (KG) has been successfully integrated into recommender systems to optimize the vector representation of entities through relational connections in the knowledge graph to improve recommendation performance [12, 13]. Knowledge graph is a heterogeneous graph, which represents the graphical structure of knowledge and information in the real world, composed of nodes and their relationships. Introducing KG into recommender systems provides several benefits [14]: (1) Leveraging the rich feature attributes of items in KG can improve recommendation performance while uncovering potential connections between items; (2) Using the different relationships in KG, the potential interests of users can be explored and the diversity of recommended items is improved.

Graph convolutional network (GCN) have garnered considerable attention in recent years, with applications in several domains [15–17]. Therefore, GCN are widely used in recommender systems [18–20]. GCN extracts synergistic information of items and users through aggregated propagation to get more suitable user-item embeddings for recommendations. Moreover, higher-order connectivity signals can be captured by overlaying multiple embedded propagation layers. Knowledge graph attention network (KGAT) [21] improves recommendation performance by training a global knowledge graph and iteratively propagating vector representations of adjacent nodes while updating the node representations through GCN. LightGCN [22] simplifies the application of GCN in recommender systems, while improving the performance of the recommender systems and reducing model complexity. The recommender methods based on GCN employ aggregation and propagation techniques to extract collaborative information between users and items. Leveraging the nodes and edges within the graph, these methods infer user interests and item attributes, thereby generating user-item embeddings that are better suited for recommendation purposes. This enhances the accuracy and efficacy of personalized recommendations. Although GCN can enhance recommendation performance, further research is needed to investigate its component methods and structures.

Despite the noteworthy successes of KG-based and GCN-based recommendation approaches, three major problems persist: (1) Solely considering the connections between users or items and entities in the knowledge graph overlooks the high-order semantic information related to the relationships between users or items and entities. (2) Focusing solely on extracting item feature information from the knowledge graph with GCN may neglect the collaborative signals between users and items. (3) The direct application of GCN's complex design elements (such as feature transformations and nonlinear activations) in recommendation leads to marginal improvements in recommendation performance and increases computational overhead [23].

Given the limitations of the aforementioned methods, it is imperative to create a method that can effectively unearth informative data from interaction graphs and item knowledge graph in an intuitive and efficient manner. This research aims to tackle this challenge by

introducing a knowledge-aware recommendations under bi-layer graph convolutional networks (BIKAGCN). The model in this paper consists of two main components:

- *Light graph convolution (LGC)*. In LGC, self-connection, feature transformations, and non-linear activation functions are removed, which simplifies GCN to a large extent and enhances its ability to extract collaborative signals from the user-item matrix.

- *KAGCN*. This research improves the aggregation layer of KGCN [24] into a knowledge-aware layer of attention and graph convolutional networks (KAGCN). It uses the attention mechanism designed in this paper to mine the importance between specific users and node relationships in the information aggregation process, by which the personalized interests of users can be captured. In addition, we incorporate a user self-attention mechanism that captures the user's differences and enhances their semantic information. The incorporation of a knowledge graph into recommendation systems is particularly crucial for addressing the needs of cold-start users. These users lack sufficient or any historical behavioral data in the system, making it challenging for traditional personalized recommendation methods to provide accurate suggestions. In recent years, a substantial body of research and methods has been dedicated to mitigating the cold-start problem. This effort ensures that even new users or new items can receive valuable recommendations, thereby enhancing the comprehensiveness and utility of recommendation systems [25–27].

In summary, our research's contributions can be outlined as follows:

1. In this paper, we propose the BIKAGCN model, which uses the first layer of GCN to learn the synergistic information of user and item, and update the user-item embeddings, and then uses the user and item embeddings as the input of the second layer of GCN.
2. By integrating bi-layer GCN to obtain the final representation of user-item embeddings, our approach effectively alleviates the cold-start problem caused by relying solely on user-item interactions and the lack of collaborative information in the item knowledge graph.
3. Experimental results on three publicly available datasets (MovieLens-20M, Last-FM, and Book-Crossing) demonstrate that BIKAGCN improved the recall@20 by 14.41%, 8.86%, and 20.90%, respectively, while the normalized discounted cumulative gain (NDCG@20) improved by 15.07%, 18.82%, and 22.79%, respectively.

2 State of the art

In this section, we review and summarize related work, including collaborative filtering-based, knowledge graph-based, and GCN-based recommendations.

Collaborative Filtering is among the earliest and widely adopted recommendation methods in predicting and suggesting new items based on user preferences [28]. MF [6] achieves good performance in recommendation by mapping user and item related features to embeddings. Moreover, FM [29] and Field-aware factorization machine (FFM) [30] utilize multiple features and blend linear combinations to enhance the performance, but they solely consider low-order feature combinations. Advanced methods must therefore integrate deep neural networks to model feature combinations for improved performance. Some distinguished models [31, 32] utilize deep neural networks and exhibit excellent recommendation performance. However, these methods only use the user-item matrix to define the loss function, failing to explicitly encode the user and item, which leads to insufficient utilization of collaborative signals.

Collaborative filtering-based recommendation methods exhibit issues involving data sparsity, cold-start, and interpretability. To address these problems, knowledge graph-aware recommendation methods have emerged, which are mainly divided into two categories: recommendation based on knowledge graph embedding (KGE) and recommendation based on knowledge graph path. The recommendation algorithm based on knowledge graph embedding utilizes the knowledge graph embedding [33] to directly link the obtained item embedding to the recommender systems, thereby improving the recommendation performance and alleviating cold-start problems. However, these KGE may prove insufficient for the purposes of recommendation, as they lack intuitiveness and effectiveness in representing item relationships. Conversely, the knowledge graph path-based method [34, 35] endeavors to explore higher-order latent entity relationships within a knowledge graph by designing specialized meta-paths or graphs that can be applied to recommendation. Although the path-based approach enhances the interpretability of the recommender systems and provides corresponding methods to explore the feature information present in the knowledge graph, the meta-paths/graphs it designs may only be applicable to specific domains and have limited generality.

LightGCN [22] presents a simplified design of a graph convolutional neural network that enhances its applicability for recommendation purposes. The recommendation model of GCN considers user and item as well as their corresponding comments [36] and interaction social networks [20] to enhance the accuracy and interpretability of recommendations. In addition, KGCN [24] uses GCN to extract the attributes between items in the knowledge graph with the aim of enhancing recommendation performance. Conversely, KGNN-LS [37] applies label smoothing to solve the overfitting problem associated with KGCN. KGAT [21] embeds a high-level knowledge graph for modeling, while the recent KLGCN [38] explores the interactions between users and relations in the knowledge graph, as well as interactions between items and relations. Although GCN-based approaches have improved recommendation performance, they still inadequately exploit the knowledge graph for personalized item-relationship mining and neglect user semantics. In this research, the user-item interaction matrix and item knowledge graph are used as inputs and relevant features are extracted by a bi-layer GCN, while accelerating model convergence and improving recommendation performance. For further comparison, we summarize our proposed model in Table 1 alongside some of the relevant methods.

The full article is organised as follows: Sect. 1 presents the current research dilemma and outlines the research methodology employed in this paper. Section 2 introduces related technologies and backgrounds. Section 3 describes the recommendation problem and presents model details and the computation process. Section 4 analyzes the experimental results, and through ablation experiments, validates the reasonableness of the model parts and the outstanding performance of the model in alleviating cold-start problems. Finally, Sect. 5 summarizes the paper and analyzes future possible improvements.

3 Methodology

3.1 Description of the Problem

In the context of recommendations, a set of k users, denoted as $m = \{u_1, u_2, \dots, u_k\}$, and a set of n items, denoted as $p = \{v_1, v_2, \dots, v_n\}$, are considered. A set $R = \{r_{u,i}\}$ represents the interactions between user and item, where $r_{u,i} = 1$ indicates the existence of interactions

Table 1 Comparison of BIKAGCN with related methods

Method	W-M	Input data		Use of technology		
		UI	KG	GNN	Att	KGE
FM [6]	✓					
NeuMF [39]	✓					
NGCF [40]	✓	✓		✓		
LightGCN [22]		✓		✓		
CKE [41]	✓		✓			✓
RippleNet [14]	✓	✓	✓			✓
KGCN [24]	✓		✓	✓	✓	
KGNN-Ls [37]	✓		✓	✓	✓	
KGAT [21]	✓	✓	✓	✓	✓	✓
KLGCN [38]		✓	✓	✓	✓	
LighterKGCN [42]	✓	✓	✓	✓		
BIKAGCN		✓	✓	✓	✓	

In the table, 'W-M' represents the weight matrix, 'UI' represents user-item interaction graph, 'GNN' represents graph neural networks, 'Att' stands for attention mechanism

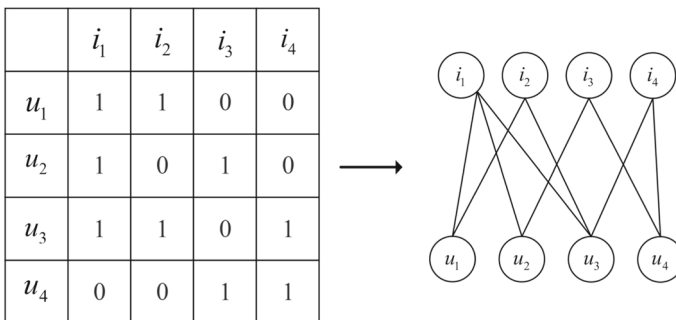


Fig. 1 A bipartite graph of a user-item constructed from the U-I matrix

between users u and items i . A user-item bipartite graph $G = \{E, V\}$ is then constructed based on the historical data of user-item interactions, as depicted in Fig. 1. Each user and item is viewed as a node of the bipartite graph, while edges with interaction records are established between user nodes and item nodes. Here, $M_{m \times n}$ refers to the adjacency matrix of the bipartite graph. The recommendation task is to predict the probability of user interaction with all items, and generally select the Top-N items with the highest predicted probability as the recommendation result.

3.2 Overall Model Description

The overall framework of the BIKAGCN model is shown in Fig. 2, which consists of three parts: an initialization layer, a bilayer graph convolution information propagation layer, and a prediction layer.

The recommender model in this research comprises several operational steps. Firstly, user nodes, item nodes, entity nodes, and relationship nodes are initialized with low-dimensional

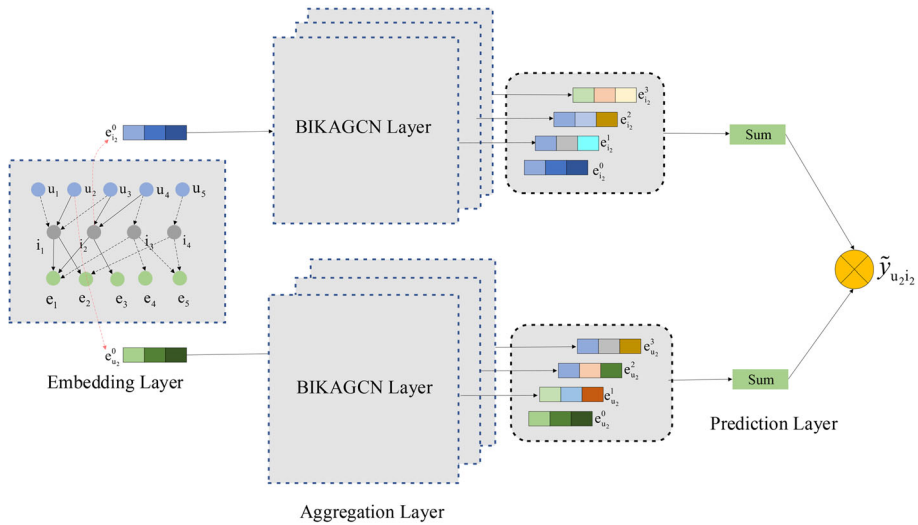


Fig. 2 The architecture of BIKAGCN comprises three primary layers: an embedding layer, a bi-layer graph convolutional information propagation layer, and a prediction layer. Figure displays a recommendation scenario between i_2 and u_2

dense embedding representations. Next, through the LGC layer, utilizing information propagation rules, the initial embedding representations of users and items are updated to extract collaborative information between them. Subsequently, the KAGCN layer aggregates the initialized entity and relationship embedding representations with the updated user-item embedding using aggregation functions to obtain entirely new user-item embedding. Unlike many GCN-based recommendation methods, the KAGCN layer employs different attention mechanisms to separately update the embeddings for users and items, thus acquiring biased user-item embedding representations. Thirdly, by stacking the aforementioned operations multiple times, multiple high-order embeddings for users and items are obtained. The purpose of this multiple stacking is to enhance the model’s representation capacity and abstraction levels, enabling it to better capture complex relationships and features in recommendations. Then, the final user-item embedding are generated by weighting these high-order embedding representations. Finally, the model employs a simple dot product between vectors to generate the ultimate recommendation prediction probabilities. The bi-layer graph convolution structure of the BIKAGCN is shown in Fig. 3.

3.3 Initial Embedding Layer

To attain the suitable, final representation, an initial embedding vector is assigned to each user denoted by $e_u \in \mathbb{R}^d$. Additionally, initial embedding vectors e_e and e_r are designated for the entities and relations, where $e_e, e_r \in \mathbb{R}^d$:

$$E = \{ \underbrace{e_{u_1}, \dots, e_{u_N}}_{\text{users embeddings}}, \underbrace{e_{i_1}, \dots, e_{i_M}}_{\text{items embeddings}}, \underbrace{e_{e_{M+1}}, \dots, e_{e_K}}_{\text{non-items/entities}}, \underbrace{e_{r_1}, \dots, e_{r_Q}}_{\text{relations embeddings}} \} \quad (1)$$

The method proposed in this paper achieves end-to-end optimization of the initial embedding. Compared with traditional methods, the method in this paper directly passes the

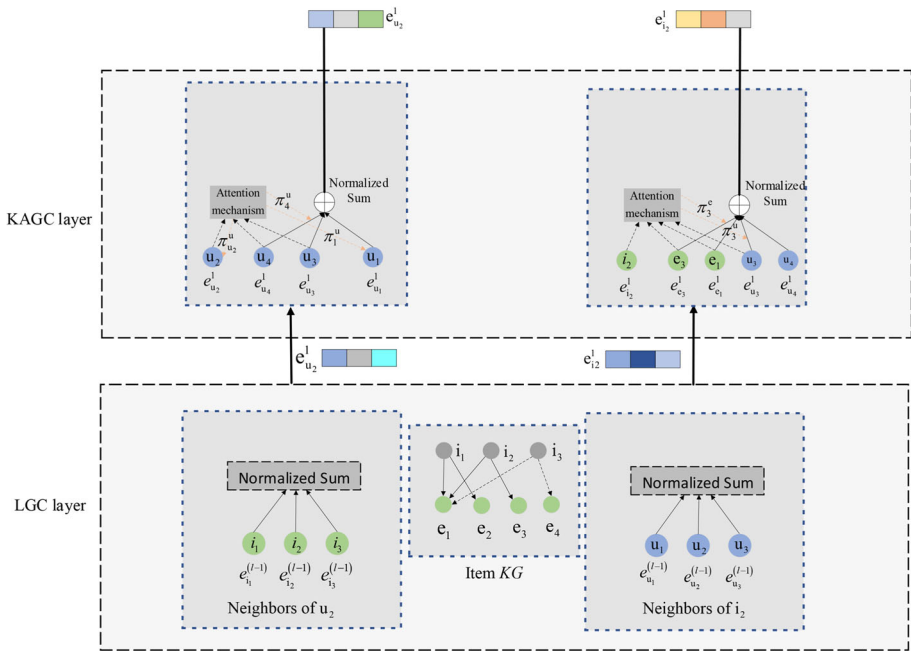


Fig. 3 An illustration of the bi-layer GCNs architecture, in the information aggregation process, first uses LGC to mine synergistic information between user-item interaction matrices and update user-item embedding representations, and then uses KAGCN to mine item knowledge graph information and update user-item embedding representations again

embedding to the information aggregation part and mines the synergistic signals in the user-item graph by LGC, thus improving the performance of the recommender systems. In addition, the problem of cold-starts was mitigated by aggregating information from the item’s knowledge graph through KAGCN.

3.4 Information Aggregation Layer

We introduce the LGC layer, followed by the KAGCN layer proposed in this paper, and we will focus on the KAGCN part.

3.4.1 The First Layer of GCN–LGC

LightGCN streamlines the architecture of the GCN model by removing the feature transformation and nonlinear activation function propagation processes. Instead, a summation aggregator generates target node embeddings at each layer utilizing the Formula 2:

$$\begin{aligned}
 e_u^{(l+1)} &= \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u| |\mathcal{N}_i|}} e_i^{(l)} \\
 e_i^{(l+1)} &= \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_u| |\mathcal{N}_i|}} e_u^{(l)}
 \end{aligned}
 \tag{2}$$

In this case, $e_u^{(l)}$ and $e_i^{(l)}$ depict the l_{th} layer embedding. $|\mathcal{N}_u|$ indicates the number of items interacted by user u . Similarly, $|\mathcal{N}_i|$ indicates the number of users interacting with item i . The goal of the symmetric normalization term $\frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}}$ is to prevent vector scale explosion.

3.4.2 The Second Layer of GCN-KAGCN

Generally, users possess distinct and individualized interests when it comes to relationships. The attention mechanism, through dynamic adjustment of recommendation results and highlighting the most relevant content to a user’s interests, can better cater to a user’s unique needs [43–46]. This, in turn, enhances the precision and personalization of recommendations. For instance, one user may be more inclined towards a movie’s genre, while another user may prioritize its cast. Consequently, our model incorporates a specifically tailored attention mechanism that takes into account each user’s unique interests during the aggregation process.

To account for the varying importance of different relations to different users, we introduce an attention function, denoted as $g : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$, as shown in Eq. (3). The attention function evaluates the relative importance of each relation that is associated with the central node, $e_u^{(l)}$ and $e_i^{(l)}$ are both outputs from LGC.

$$\begin{aligned} \omega_{rei}^u &= g(e_u^{(l)}, r_{ei}) \\ \omega_{rei}^i &= g(e_i^{(l)}, r_{ei}) \\ \omega_{rei}^{ui} &= \alpha \omega_{rei}^u + \beta \omega_{rei}^i \end{aligned} \tag{3}$$

where r_{ei} represents the relationship between entity e ($e \in \mathcal{N}_i$) and item i , and \mathcal{N}_i represents the neighboring entities of item i in the knowledge graph. We use a strategy similar to GraphSage [47] to maintain computational efficiency and rigidity due to the large number of entities and connections in real-world knowledge graph. Specifically, we use a technique where uniform and randomly sampled fixed-size neighbor nodes K are employed as local neighbors in place of full neighbors. This results in a new set of neighbors, S_i , whose size is K , with the potential for duplicate entries when $|\mathcal{N}_i| < K$. The final attention score is calculated by taking the sum of the attention weights ω_{rei}^u and ω_{rei}^i , and setting $\alpha = \beta = 0.5$ in the experiment.

We generate representations of the target entity e_{S_i} by aggregating neighbor embeddings, as shown in Formula 4.

$$e_{S_i}^{(l+1)} = \sum_{e \in S_i} \tilde{\omega}_{rei}^{ui} e_e^{(l)} \tag{4}$$

Where $\tilde{\omega}_{re}^{ui}$ represents the normalized use-item relational attention score, which indicates that the target node will value neighbouring nodes with higher relational attention scores, where $\tilde{\omega}_{re}^{ui}$ is defined by the following Formula (5):

$$\tilde{\omega}_{rei}^{ui} = \frac{\exp(\omega_{rei}^{ui})}{\sum_{e' \in S_i} \exp(\omega_{re'}^{ui})} \tag{5}$$

The self-attention mechanism stands as a widely applied technique in deep learning an natural language processing [48]. Its fundamental concept revolves around the allocation of weights to each position within an input sequence based on their respective importance and subsequently amalgamating the information from all positions. Self-attention mechanism has

the capacity to capture interdependencies among data, facilitating enhanced user perception and information fusion. This paper adds a self-attention mechanism on the user side. Firstly, according to the user embedding calculation:

$$\begin{aligned} Q_u^{(l)} &= W^q \cdot e_u^{(l)} \\ K_u^{(l)} &= W^k \cdot e_u^{(l)} \\ V_u^{(l)} &= W^v \cdot e_u^{(l)} \end{aligned} \tag{6}$$

In this step, the attention score for the $Q_u^{(l)}$ query and $K_u^{(l)}$ is calculated.

$$\omega_{u'u}^u = F(Q_u^{(l)}, K_u^{(l)}) \tag{7}$$

The next step involves calculating the similarity of $Q_u^{(l)}$ with $K_u^{(l)}$, which is usually achieved by utilizing the softmax function in the attention mechanism.

$$\tilde{\omega}_{u'u}^u = \frac{\exp(\omega_{u'u}^u)}{\sum_{u' \in S_u} \exp(\omega_{u'u}^u)} \tag{8}$$

The final computed attention weights are weighted and summed over $V_u^{(l)}$.

$$e_{S_u}^{(l+1)} = \sum_{u' \in S_u} \tilde{\omega}_{u'u}^u V_u^{(l)} \tag{9}$$

3.4.3 Aggregation Functions

The phase of information propagation comprises the process of aggregating entity representations $e_i^{(l)}$ and their corresponding domain representations $e_{S_i}^{(l)}$ to obtain a novel entity representation l . In order to integrate additional domain semantic information, we explore two approaches to aggregate $e_i^{(l)}$ and $e_{S_i}^{(l)}$ to obtain the ultimate entity representation:

(1) GCN-s aggregator. We remove the feature transformation and non-linear activation function of the original GCN aggregator [49] to obtain the final entity representation, as shown in Formula (10):

$$f_{GCN-s} = e_i^{(l)} + e_{S_i}^{(l)} \tag{10}$$

(2) Neighbor-s aggregator. We remove the feature transformation and non-linear activation function of the original Neighbor aggregator [50] to get the final entity representation. As shown in Formula (11):

$$f_{Neighbor-s} = e_{S_i}^{(l)} \tag{11}$$

In Algorithm 1, we summarize the entire flow of KAGCN, including the inputs and outputs, as well as the intermediate steps.

3.5 Prediction Layer

After stacking the bi-layer GCN L times, vector representations of $L+1$ item ($\{e_i^{(1)}, \dots, e_i^{(L)}\}$) are obtained, as well as those of $L+1$ users ($\{e_u^{(1)}, \dots, e_u^{(L)}\}$). The embeddings of the L layers are merged in a weighted sum to obtain the final vector representations of user and item, as

Algorithm 1 Embedding generation process of **KAGCN**

Require: a user-item pair (u, i) ; item KG \mathbf{G} ; LGC embeddings \mathbf{E} ; hyperparameters: layer L , sampled neighbor size K ;

Ensure: the user embedding e_u^{KAGCN} ; the item embedding e_i^{KAGCN} ;

```

1:  $\{\mathcal{N}_i^l\}_{l=0}^L, \{\mathcal{N}_u^l\}_{l=0}^L \leftarrow \text{GetNeighbors}(i(u), \mathbf{G}, L, K)$   $\triangleright$  generate H-hop neighbors of user  $u$  and item  $i$ ;
2: for  $l = 1, 2, \dots, L$  do
3:   for  $e$  in  $\mathcal{N}_i^l$  do
4:      $e_{S_e}^{(l)} = \sum_{e' \in S_e} \tilde{\omega}_r^{ue'} e_e^{(l-1)}$   $\triangleright$  aggregate the (L-1)-hop neighbor features of item  $i$ ;
5:      $e_{S_u}^{(l)} = \sum_{u' \in S_u} \tilde{\omega}_r^{u'u} V_u^{(l-1)}$   $\triangleright$  aggregate the (L-1)-hop neighbor features of user  $u$ ;
6:   end for
7:    $e_i^{(l)} = \text{agg}(e_i^{(l-1)}, e_{S_i}^{(l)})$   $\triangleright$  agg represents an aggregation function[3.4.3];
8:    $e_u^{(l)} = \text{agg}(e_u^{(l-1)}, e_{S_u}^{(l)})$   $\triangleright$  agg represents an aggregation function[3.4.3];
9: end for
10:  $e_i^{KAGCN(l)} = e_i^{(l)}$   $\triangleright$  get KAGCN embedding of item  $i$ ;
11:  $e_u^{KAGCN(l)} = e_u^{(l)}$   $\triangleright$  get KAGCN embedding of user  $u$ ;
12: return  $e_u^{KAGCN}, e_i^{KAGCN}$ 
13:
14: function GetNeighbors  $(i(u), \mathbf{G}, L, K)$ 
15:    $\mathcal{N}_i^L = i$ 
16:   for  $l = L - 1, \dots, 0$  do
17:      $\mathcal{N}_i^l = \emptyset$ 
18:     for  $e$  in  $\mathcal{N}_i^{l+1}$  do
19:        $\mathcal{N}_e \leftarrow e, \mathbf{G}$   $\triangleright$  get neighbors of entity  $e$ ;
20:        $S_e \leftarrow K, \mathcal{N}_e$   $\triangleright$  sample  $K$  neighbors from the entire neighbors;
21:        $\mathcal{N}_i^l = \mathcal{N}_i^l \cup S_e$   $\triangleright$  update the (L-1)-hop neighbors of item  $i$ ;
22:     end for
23:   end for
24:   return  $\{\mathcal{N}_i^l\}_{l=0}^L$ 

```

shown in the following Formula (12):

$$\begin{aligned}
 e_u^* &= \sum_{l=0}^L \alpha_l e_u^{BIKAGCN(l)} \\
 e_i^* &= \sum_{l=0}^L \alpha_l e_i^{BIKAGCN(l)}
 \end{aligned}
 \tag{12}$$

Where, α_l represents the weight of the l_{th} layer vector in generating the final vector, and in the experiment it is set as a constant $1 / (l + 1)$.

Finally, we perform inner product operations on the representations of the user and the item to predict the scores of their interactions:

$$\hat{y}(u, i) = e_u^{*T} e_i^*
 \tag{13}$$

The expression $\hat{y}(u, i)$ represents the probability that the user u will interact with the item i . When the probability is greater than or equal to 0.5, we consider that the user will interact with the item.

3.6 Loss Function

The BIKAGCN model includes a large number of training parameters, leading to a potential overfitting problem. To alleviate this problem, L_2 regularization techniques can be used to some extent.

The network parameters of the whole model are learned with the help of the BPR loss function [51]. The loss function is defined as follows:

$$\mathcal{L} = - \sum_{(u,i,j) \in \mathcal{O}} \ln \sigma (\hat{y}_{ui} - \hat{y}_{uj}) + \lambda \|\Theta\|_2^2 \tag{14}$$

where $(u, i) \in R^+$ represents positive samples and $(u, j) \in R^-$ represents negative samples, where Θ is the set of all model parameters and $\lambda \|\Theta\|_2^2$ represents the normalized term of L_2 .

To describe the computational process of BIKAGCN visually, it is summarized in Algorithm 2.

Algorithm 2 The overall recommendation process of BIKAGCN

Require: a user-item pair (u, i) ; User interaction matrix with item Y ; item KG G ; initial embeddings E ; hyperparameters: layer L , sampled neighbor size K , $\{\alpha_l\}_{l=0}^L$;

Ensure: Predictive Function $F(u, i | Y, G, \theta)$;

- 1: **while** BIKAGCN not converge **do**
 - 2: **for** $l = 1, 2, \dots, L$ **do**
 - 3: **for** (u, i) in Y **do**
 - 4: $e_u^{LGC(l)}, e_i^{LGC(l)} = LGC((u, i), Y, E, L, \{\alpha_l\}_{l=0}^L)$ \triangleright The L -layer embedding of the LGC generated by the algorithm LGC[22];
 - 5: $e_u^{BIKAGCN(l)}, e_i^{BIKAGCN(l)} = KAGCN((u, i), G, (e_u^{LGC(l)}, e_i^{LGC(l)}), L, K)$ \triangleright Generate new user-item embeddings using the algorithm 1;
 - 6: **end for**
 - 7: **end for**
 - 8: $e_u^* = \sum_{l=0}^L \alpha_l e_u^{BIKAGCN(l)}$ \triangleright generate the final BIKAGCN embedding of user u ;
 - 9: $e_i^* = \sum_{l=0}^L \alpha_l e_i^{BIKAGCN(l)}$ \triangleright generate the final BIKAGCN embedding of item i ;
 - 10: Calculate predicted probability $\hat{y}(u, i) = e_u^{*T} e_i^*$;
 - 11: Update parameters, i.e., initial embeddings E , by gradient descent;
 - 12: **end while**
 - 13: **return** F ;
-

4 Experimentation and analysis

This section presents the recommended performance of BIKAGCN in three real recommendation scenarios (MovieLens-20M, Last-FM, and Book-Crossing), including comparative experimental performance, performance to mitigate the cold-start problem, and ablation experiments.

4.1 Experimental Data and Metrics

The performance of all models was evaluated on three real recommendation datasets (MovieLens-20M, Book-Crossing and Last-FM). These datasets have widespread availability and are extensively employed by researchers. With their variability in both size and sparsity,

Table 2 Statistics of datasets

	MovieLens-20M	Last-FM	Book-crossing
User	124,649	1251	1248
Item	16,910	3846	14965
Interactions	7,814,478	16,669	39,755
Sparsity	0.9963	0.9979	0.9965
Entities	102,569	9366	77,903
Relations	32	60	25
KG triples	3,399,048	58,554	367,478

these datasets are deemed appropriate for conducting comprehensive and robust evaluations of the performance of BIKAGCN across varying dataset sizes.

*MovieLens-20M*¹ is a widely used benchmark dataset. It includes nearly 20 million explicit historical rating records on the MovieLens website.

*Last-FM*² consists of nearly 93,000 musician listens from 2,000 users of the Last.fm online music system.

*Book-crossing*³ is a dataset containing book ratings and reviews. This dataset contains 1 million book item rating records.

In view of the datasets' explicit feedback nature, we converted them into implicit feedback by categorizing positive ratings as 1 and negative ratings as 0. For the MovieLens-20M dataset, we considered ratings of 4 or higher as positive, while for the other datasets, we did not set any specific threshold. Our selection criterion for each dataset required the inclusion of users with a minimum of ten interactions to ensure data quality.

In addition to the records of user-item interactions, we also require a corresponding item knowledge graph. We utilize project knowledge graphs provided by KGCN [24], which have been constructed using Microsoft Satori.⁴ The summary statistics of the three datasets and the knowledge graphs associated with them are shown in Table 2.

Recall: the proportion of correctly recommended Top-N items that are of real interest to the user out of the total number of items that the user interacted with, with the formula shown in Formula (15):

$$recall@k = \frac{\sum_{u \in U} |R(u) \cap T(u)|}{\sum_{u \in U} |T(u)|} \quad (15)$$

where $R(u)$ represents the element recommendation list generated by the model for user u , while $T(u)$ denotes the set of elements with which user u has genuinely interacted in the test dataset.

NDGC: It not only considers the percentage of correctly recommended items but also takes into account the position of recommended items that are of interest to the user within the recommendation list. A higher numerical value indicates better performance, as shown in Formula (16).

$$ndcg@k = \frac{1}{n} \sum_{i=1}^N \frac{2^{rel_i} - 1}{\log_2(i + 1)} \quad (16)$$

¹ <https://grouplens.org/datasets/movielens/>

² <https://grouplens.org/datasets/hetrec-2011>

³ <http://www2.informatik.uni-freiburg.de/~chiegler/BX/>

⁴ <https://searchengineland.com/library/bing/bing-satori>

rel_i represents the user's rating for the i -th item, and $\log_2(i + 1)$ is a position-dependent decreasing weight.

In this paper, our dataset will be partitioned into a train set and a test set in an 8:2 ratio. To generate negative samples, items without any positive ratings will be chosen at random and paired with each positive interaction. Our metrics will be recall@N and ndcg@N to assess the performance of the item recommendation and user preference ranking models. To determine the hyperparameters, we will optimize recall@20 on the test set, with a default value of N=20.

4.2 Baselines

We compared BIKAGCN with the following baselines on the same dataset and metrics to demonstrate its superiority. The details are as follows:

- *MF* [6]: The model is a common recommender systems algorithm that performs recommendations by decomposing the user-item interaction matrix into two low-rank matrices.
- *CKE* [41]: The model is a typical regularisation-based approach that uses semantic embeddings obtained from TransR to enhance the matrix decomposition.
- *NGCF* [40]: The model uses standard GCN to capture synergistic signals from user-item interaction graphs, effectively placing synergistic signals in the representation of user and item.
- *KGCN* [24]: The model is a recommender systems algorithm based on knowledge graph, which embeds the entities and relationships in the knowledge graph into the recommendation model to improve the accuracy and effectiveness of recommendations.
- *KGAT* [21]: The model alternates between training recommendations and knowledge graph embedding. All structural and semantic information is obtained from a combination of the user-item graph and the item knowledge graph.
- *LightGCN* [22]: The model is an efficient GCN-based model that abandons the feature transformation and non-linear activation operations in the GCN that are not useful for recommendations.

4.3 Hyperparameters Settings

Hyperparameters play a crucial role in influencing the results of the experiment. For the embedding size, we analyzed all models within a given dataset using a fixed size 64. We initialized trainable parameters using the Xavier initialization method. To avoid local optimization issues during training, we employed the mini-batch Adam optimizer. The batch size for MovieLens-20M was set to 2048, while for other datasets, it was set to 1024. We fine-tuned the following parameters with grid search: the learning rate was tested within a range of {0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001}; the L_2 regularization factor, λ , was varied between 10^{-1} and 10^{-5} ; the number of layers in GCN recommendation models (KGCN, KGAT, NGCF, LightGCN, and BIKAGCN) was adjusted between {1, 2, 3, 4} for layer size, and between {4, 8, 16, 32} for neighbor sampling size in KGCN and BIKAGCN. For KGCN, the hidden dimension is set to the same size as the initial embedding dimension, and for KGAT and NGCF, the size of the first hidden dimension is the same as the initial embedding dimension, but the subsequent hidden dimension is half of the previous one. KGAT and NGCF use a 0.1 dropout ratio. Furthermore, owing to the vast scale of the MovieLens-20M dataset, we deployed an early stopping, whereby if the recall@20 metric on the test set did not exhibit any progress within ten epochs, we resorted to early stopping.

Table 3 Comparison of the recommended performance with the baseline method

	MovieLens-20M		Last-FM		Book-crossing	
	recall@20	ndcg@20	recall@20	ndcg@20	recall@20	ndcg@20
MF	0.3392	0.2820	0.3742	0.2303	0.0978	0.0756
CKE	0.3321	0.2756	0.3882	0.2362	0.0955	0.0753
KGCN	0.3352	0.2794	0.3843	0.2199	0.0998	0.0787
KGAT	0.3419	0.2900	0.3924	0.2411	0.1190	0.0904
NGCF	0.3389	0.2833	0.3954	0.2417	0.1001	0.0829
LightGCN	0.3643	0.2914	0.4264	0.2587	0.1450	0.1009
BIKAGCN	0.4168	0.3353	0.4642	0.3074	0.1753	0.1239

Bold values indicate the emphasize the experimental results of our model

4.4 Experimental Results

4.4.1 Comparison of Recommended Performance with Baseline

From Table 3, we can conclude the following:

Among all the recommendation models based on graph neural networks, KGCN exhibits the weakest performance across all three datasets. This could be attributed to KGCN producing more links between items while overlooking the links between user and item. Additionally, KGCN's large number of trainable parameters may cause overfitting, diminishing its generalization ability and restricting its performance.

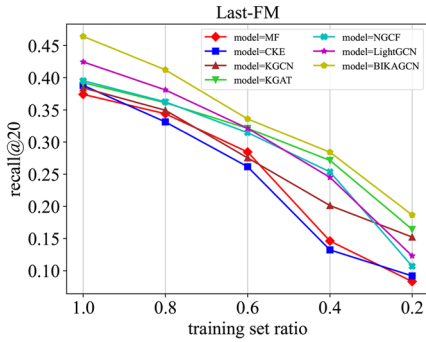
In comparison with NGCF, the KGCN model's performance was relatively poorer across the three datasets, indicating the limited strength of semantic information obtained exclusively from the item knowledge graph and emphasizing the significance of synergistic information between user and item. When comparing NGCF and LightGCN, the latter exhibiting enhanced recommendation performance across all three datasets, emphasizing the advantages of eliminating feature transformation and nonlinear activation in recommendations. Furthermore, BIKAGCN performs better than KGAT across all three datasets, implying that it is not essential to train the entire knowledge graph simultaneously for recommendation purposes.

A comparative analysis of all baselines showed that BIKAGCN has a clear advantage on all datasets. This suggests that BIKAGCN's bilayer graph convolutional neural network layer possesses the ability of LGC to extract synergistic information and KAGCN to selectively and preferentially aggregate item neighbourhood information. BIKAGCN is adept at merging complementary information from item knowledge graph into item vector representations and gaining insight into the unique requirements and potential interests of different users; as a result, it produces better recommendations, thereby improving the performance of the recommender systems.

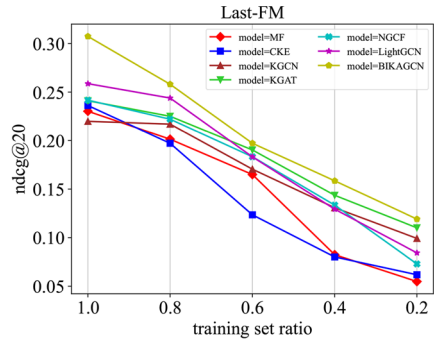
4.4.2 Comparison of Model Cold-Start Scenarios

By conducting a joint analysis of the experimental results from Last-FM and Book-Crossing, we can derive some meaningful conclusions, which are presented in Fig. 4.

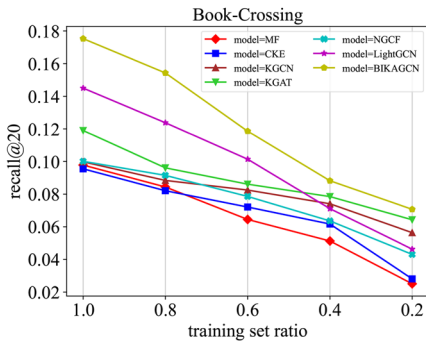
In scenarios where data is sparse, adding the item KG as additional information into the recommender system can mitigate the cold-start problem, especially in cases of extreme data scarcity. When the training set ratio drops to 0.2, three recommendation models based on



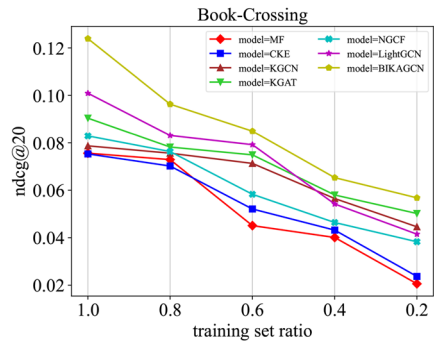
(a) recall@20 on Last-FM



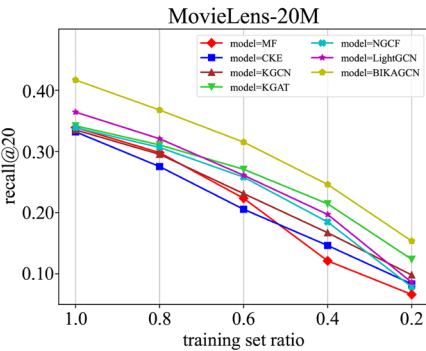
(b) ndcg@20 on Last-FM



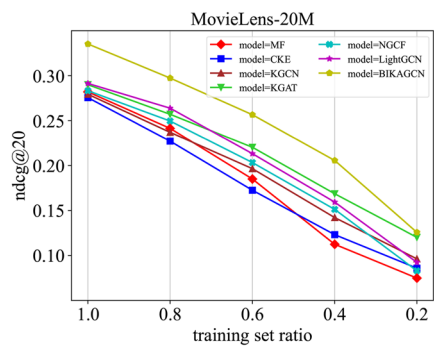
(c) recall@20 on Book-Crossing



(d) ndcg@20 on Book-Crossing



(e) recall@20 on MovieLens-20M



(f) ndcg@20 on MovieLens-20M

Fig. 4 Performance comparison of training sets with different scales on the three datasets. The training set ratio indicates the ratio of the current training set to the original training set

GCN and KG (BIKAGCN, KGAT, and KGCN) perform the best, indicating the effectiveness of incorporating knowledge graphs in alleviating the cold-start problem. Possible reasons are that, in cold-start scenarios, the KG provides additional high-order item correlations and preference relationship information between users and items. Introducing this information can mitigate data sparsity, compensate for the lack of item information, and maintain the performance of the recommender systems.

Table 4 Performance of BIKAGCN with different aggregation functions

	MovieLens-20M		Last-FM		Book-crossing	
	recall@20	ndcg@20	recall@20	ndcg@20	recall@20	ndcg@20
GCN	0.3714	0.2931	0.4208	0.2735	0.1639	0.1186
GCN-s	0.4168	0.3353	0.4642	0.3074	0.1753	0.1239
Neighbor	0.3556	0.2871	0.3817	0.2572	0.1351	0.0932
Neighbor-s	0.3744	0.3091	0.3975	0.2627	0.1438	0.0955

Bold value indicate the optimal results under different experimental conditions

Table 5 Performance of BIKAGCN with different layer number

	MovieLens-20M		Last-FM		Book-crossing	
	recall@20	ndcg@20	recall@20	ndcg@20	recall@20	ndcg@20
<i>layers_L</i> = 1	0.4168	0.3353	0.4245	0.2852	0.1569	0.1172
<i>layers_L</i> = 2	0.4015	0.3299	0.4642	0.3074	0.1753	0.1239
<i>layers_L</i> = 3	0.3860	0.3353	0.4108	0.2716	0.1714	0.1271
<i>layers_L</i> = 4	0.3445	0.2897	0.4090	0.2611	0.1632	0.1215

Bold value indicate the optimal results under different experimental conditions

BIKAGCN excels in cold-start scenarios. The recommendation results across three datasets demonstrate that it outperforms other models at all levels of sparsity. This suggests that BIKAGCN can effectively address the cold-start problem, leading to the conclusion that the model is a reliable solution for the cold-start problem.

4.5 Ablation Study of BIKAGCN

The graphical convolution layer is a key component of the GCN-based model. The effects of different aggregation functions, different aggregation depths, different sampling neighborhoods and different attention mechanisms on the model performance are explored.

4.5.1 Impact of BIKAGCN Aggregators

Table 4 summarizes the results of our experimental evaluation of four aggregation functions, from these findings, several noteworthy conclusions can be drawn:

The results of our experiments comparing GCN-s with GCN and Neighbor-s with Neighbor have demonstrated that the elimination of feature transformation and non-linear activation functions contributes positively to enhancing the performance of recommender systems. Representing item features solely via aggregated vectors of neighboring nodes is insufficient for fully exploiting user and item information. Notably, we have found that incorporating self-connection to item features can further boost the performance of the recommender systems.

4.5.2 Impact of BIKAGCN Layer Number

To explore the impact of the number of layers on performance, analyzing the Table 5 we have the following observations:

Table 6 Performance of BIKAGCN with sampled neighbor number

	MovieLens-20M		Last-FM		Book-crossing	
	recall@20	ndcg@20	recall@20	ndcg@20	recall@20	ndcg@20
$neighbor_K = 4$	0.3801	0.3129	0.3850	0.2228	0.1392	0.0983
$neighbor_K = 8$	0.4168	0.3353	0.4642	0.3074	0.1548	0.1096
$neighbor_K = 16$	0.3954	0.3183	0.4075	0.2642	0.1753	0.1239
$neighbor_K = 32$	0.3638	0.3031	0.4092	0.2689	0.1431	0.0934

Bold value indicate the optimal results under different experimental conditions

In all three datasets, the best performance was achieved at $layers_L$ of 1 or 2. The reason is that increasing the number of layers enhances the amount of surrounding neighbor information that can be acquired, which leads to a richer information context and improves the performance of the recommender systems. It is evident that increasing the number of propagation layers does not necessarily enhance the performance of the recommender systems model. This is because deeper layers may lead to the problem of overly smooth neighboring nodes, i.e., the vector representations of nodes become increasingly similar, resulting in the inability of the network to identify items of interest to users. Moreover, deeper layers also mean that more negative information is aggregated. Therefore, it is necessary to select an appropriate number of layers based on practical requirements.

4.5.3 Impact of BIKAGCN Sampled Neighbor Number

Table 6 shows the results of changing the size of the neighbors, and we can find that:

When K equals 8 or 16, BIKAGCN achieves the best performance. In the MovieLens-20M and Last-FM datasets, the best number of neighbors is 8, whereas in the Book-Crossing dataset, the optimal number of neighbors is 16. This is because the MovieLens-20M dataset is large enough to obtain sufficient semantic information with 8 neighbors. In contrast, increasing the number of neighbors can introduce more noise. In the Last-FM dataset, the model can be easily fitted due to its small size, and a large number of neighbors can introduce excessive noise. For the Book-Crossing dataset, which is both sparse and small, the optimal number of neighbors is 16. Beyond the neighbor threshold, the performance of BIKAGCN rapidly decreases as the number of neighbors increases, indicating that more item noise is aggregated, and the negative impact gradually exceeds the positive impact.

4.5.4 Impact of BIKAGCN Attention Mechanism

The attention mechanism is a critical factor that significantly influences model performance. Table 7 presents the experimental results, from which several noteworthy observations can be made.

In most cases, the performance of the recommender systems improves with an increase in the number of attention mechanisms, except for $ndcg@20$ on the MovieLens-20M dataset. Additionally, "ur+ir+self" consistently yields the best model performance. Notably, using "ur" mechanisms outperforms "ir" on in all cases, which underscores the indispensable role of personalized interest in recommendation and its importance exceeding that of neighbors.

Comparison between "ur+ir+self" and "ur+ir" reveals that the former outperforms the latter. One possible reason for this superior performance is that considering user attention

Table 7 Performance of BIKAGCN with attention mechanism

	MovieLens-20M		Last-FM		Book-Crossing	
	recall@20	ndcg@20	recall@20	ndcg@20	recall@20	ndcg@20
no ¹	0.3456	0.2861	0.3850	0.2155	0.1291	0.0934
ur ²	0.3775	0.3088	0.3975	0.2243	0.1533	0.1154
ir ³	0.3649	0.2943	0.3887	0.2192	0.1428	0.1021
ur+ir ⁴	0.3786	0.3071	0.4223	0.2587	0.1674	0.1198
ur+ir+self ⁵	0.4168	0.3353	0.4642	0.3074	0.1753	0.1239

¹ Not using attention mechanism

² Only using user relation attention mechanism

³ Only using item relation attention mechanism

⁴ Using both user and item relation attention mechanisms at the same time

⁵ Using user relation and item relation attention along with user self-attention mechanism

Bold value indicate the optimal results under different experimental conditions

enables the model to leverage user semantic information fully and enhance the performance of the recommender systems. On the other hand, the reason "ur+ir" outperforms "ur" may be that the former thoroughly explores the connections and relationships between user and item, thus promoting the aggregation of item neighboring nodes and improving the recommendation algorithm's performance. However, when the result without an attention mechanism is used (i.e., using "no"), it is shown that treating all neighbors equally introduces noise and misleads the embedding propagation process, indicating the important role of graph attention mechanism.

5 Conclusions

In this paper, we propose a new model—BIKAGCN, which combines the respective characteristic features of LGC and KAGCN. Specifically, LightGCN is used as the first layer of the model, followed by KAGCN as the second layer, resulting in the integration of two into a novel bi-layer graph convolutional network. This innovative combination enhances the feature extraction capability of GCN and accelerates model convergence. Our experimental results conducted on three real-world datasets demonstrate that incorporating a GCN layer after LightGCN to learn the embedding representations of users and entities in the knowledge graph is both effective and feasible, thereby providing insights for the development of graph neural-based recommender systems in the future.

We identify two key areas for future research. Firstly, regarding the negative sampling strategy in the model, there are several studies exploring ways to improve upon the current uniform sampling technique [20, 52–55]. One potential approach involves sampling a different number of neighboring entities for each item, which may result in enhanced recommender system performance. Secondly, our experimental results suggest that incorporating item KG as auxiliary information can improve performance compared to using only the interaction graph. Thus, research could examine the effectiveness of incorporating additional types of auxiliary data, such as contextual data [56] or social networks [57]. In social network recommendation, identifying reliable friends is also a popular research direction [58, 59].

Acknowledgements This work is supported by Sichuan Science and Technology Program (2022NSFSC0555).

Author Contributions GL: Conceptualization, methodology, validation, formal analysis, investigation, data organization, drafting, visualization. YL: Validation, writing-reviewing and editing, project management, fundraising. SB and XS: editing and proofreading. YR and SL: reviewing.

Funding Sichuan Science and Technology Program (2022NSFSC0555)

Data Availability The MovieLens-20M data supporting the findings of this study are available at <https://grouplens.org/datasets/movielens/>, Last-FM data are available from <https://grouplens.org/datasets/hetrec-2011> and Book-Crossing data are available from <http://www2.informatik.uni-freiburg.de/~chiegler/BX/>.

Declarations

Conflict of 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.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Javed U, Shaukat K, Hameed IA, Iqbal F, Alam TM, Luo S (2021) A review of content-based and context-based recommendation systems. *Int J Emerg Technol Learn* 16(3):274–306
2. Tahmasebi H, Ravanmehr R, Mohamadzaei R (2021) Social movie recommender system based on deep autoencoder network using twitter data. *Neural Comput Appl* 33:1607–1623
3. Xiao Z, Yang L, Jiang W, Wei Y, Hu Y, Wang H (2020) Deep multi-interest network for click-through rate prediction. In: *Proceedings of the 29th ACM international conference on information & knowledge management*, pp 2265–2268
4. 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
5. 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
6. Koren Y, Bell R, Volinsky C (2009) Matrix factorization techniques for recommender systems. *Computer* 42(8):30–37
7. Mandal S, Maiti A (2018) Explicit feedbacks meet with implicit feedbacks: a combined approach for recommendation system. In *International conference on complex networks and their applications*, Springer, pp 169–181
8. Park C, Kim D, Oh J, Yu H (2017) Do "also-viewed" products help user rating prediction? In: *Proceedings of the 26th international conference on world wide web*, pp 1113–1122
9. Mandal S, Maiti A (2020) Explicit feedback meet with implicit feedback in gpmf: a generalized probabilistic matrix factorization model for recommendation. *Appl Intell* 50(6):1955–1978
10. Resnick P, Varian HR (1997) Recommender systems. *Commun ACM* 40(3):56–58
11. Mandal S, Maiti A (2021) Rating prediction with review network feedback: a new direction in recommendation. *IEEE Trans Comput Soc Syst* 9(3):740–750
12. Yang Z, Dong S (2020) Hagerrec: hierarchical attention graph convolutional network incorporating knowledge graph for explainable recommendation. *Knowl Based Syst* 204:106194
13. Sang L, Xu M, Qian S, Wu X (2021) Knowledge graph enhanced neural collaborative recommendation. *Expert Syst Appl* 164:113992

14. Wang H, Zhang F, Wang J, Zhao M, Li W, Xie X, Guo M (2018) Ripplenet: propagating user preferences on the knowledge graph for recommender systems. In: Proceedings of the 27th ACM international conference on information and knowledge management, pp 417–426
15. Ying R, He R, Chen K, Eksombatchai P, Hamilton WL, 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
16. Xu Y, Jin S, Chen Z, Xie X, Hu S, Xie Z (2022) Application of a graph convolutional network with visual and semantic features to classify urban scenes. *Int J Geograph Inform Sci* 36(10):2009–2034
17. Zeb A, Saif S, Chen J, Haq AU, Gong Z, Zhang D (2022) Complex graph convolutional network for link prediction in knowledge graphs. *Expert Syst Appl* 200:116796
18. Mandal S, Maiti A (2021) Graph neural networks for heterogeneous trust based social recommendation. In: 2021 international joint conference on neural networks (IJCNN), IEEE, pp 1–8
19. Rossi R.A, Ahmed N.K, Koh E (2018) Higher-order network representation learning. In: Companion proceedings of the web conference 2018, pp 3–4
20. Fan W, Ma Y, Li Q, He Y, Zhao E, Tang J, Yin D (2019) Graph neural networks for social recommendation. In: The world wide web conference, pp 417–426
21. 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
22. 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
23. Lu W, Altenbek G (2021) A recommendation algorithm based on fine-grained feature analysis. *Expert Syst Appl* 163:113759
24. 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
25. Wei J, He J, Chen K, Zhou Y, Tang Z (2017) Collaborative filtering and deep learning based recommendation system for cold start items. *Expert Syst Appl* 69:29–39
26. Anwaar F, Iltaf N, Afzal H, Nawaz R (2018) Hrs-ce: a hybrid framework to integrate content embeddings in recommender systems for cold start items. *J Comput Sci* 29:9–18
27. Vartak M, Thiagarajan A, Miranda C, Bratman J, Larochelle H (2017) A meta-learning perspective on cold-start recommendations for items. *Adv Neural Inform Process Syst* 30
28. Su X, Khoshgoftaar TM (2009) A survey of collaborative filtering techniques. *Adv Artif Intell* 2009:1–9
29. Rendle S (2010) Factorization machines. In: 2010 IEEE international conference on data mining, IEEE, pp 995–1000
30. 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
31. Cheng H.-T, Koc L, Harmsen J, Shaked T, Chandra T, Aradhye H, Anderson G, Corrado G, Chai W, Isipir M, et al (2016) Wide & deep learning for recommender systems. In: Proceedings of the 1st workshop on deep learning for recommender systems, pp 7–10
32. 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
33. Yang B, Yih W.-T, He X, Gao J, Deng L (2014) Embedding entities and relations for learning and inference in knowledge bases. [arXiv:1412.6575](https://arxiv.org/abs/1412.6575)
34. Palumbo E, Monti D, Rizzo G, Troncy R, Baralis E (2020) entity2rec: property-specific knowledge graph embeddings for item recommendation. *Expert Syst Appl* 151:113235
35. 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, vol 33, pp 5329–5336
36. Wu L, Quan C, Li C, Wang Q, Zheng B, Luo X (2019) A context-aware user-item representation learning for item recommendation. *ACM Trans Inform Syst* 37(2):1–29
37. Wang H, Zhang F, Zhang M, Leskovec J, Zhao M, Li W, Wang Z (2019) Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, pp. 968–977
38. Wang F, Li Y, Zhang Y, Wei D (2022) Klgn: knowledge graph-aware light graph convolutional network for recommender systems. *Expert Syst Appl* 195:116513
39. He X, Liao L, Zhang H, Nie L, Hu X, Chua T.-S (2017) Neural collaborative filtering. [arXiv:1708.05031](https://arxiv.org/abs/1708.05031)
40. 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

41. 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
42. Chen P, Zhao J, Yu X (2022) Lighterkgcn: a recommender system model based on bi-layer graph convolutional networks. *J Int Technol* 23(3):621–629
43. Dong X, Yu L, Wu Z, Sun Y, Yuan L, Zhang F (2017) A hybrid collaborative filtering model with deep structure for recommender systems. In: Proceedings of the AAAI conference on artificial intelligence, vol 31
44. Mandal S, Maiti A (2022) Fusiondeepmf: A dual embedding based deep fusion model for recommendation. [arXiv:2210.05338](https://arxiv.org/abs/2210.05338)
45. Yin H, Wang W, Wang H, Chen L, Zhou X (2017) Spatial-aware hierarchical collaborative deep learning for poi recommendation. *IEEE Trans Knowl Data Eng* 29(11):2537–2551
46. Mandal S, Maiti A (2021) Deep collaborative filtering with social promoter score-based user-item interaction: a new perspective in recommendation. *Appl Intell* 51:7855–7880
47. Hamilton W, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. *Adv Neural Inform Process Syst* 30
48. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A.N, Kaiser Ł, Polosukhin I (2017) Attention is all you need. *Adv Neural Inform Process Syst* 30
49. Kipf T.N, Welling M (2016) Semi-supervised classification with graph convolutional networks. [arXiv:1609.02907](https://arxiv.org/abs/1609.02907)
50. Veličković P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y (2017) Graph attention networks. [arXiv:1710.10903](https://arxiv.org/abs/1710.10903)
51. Rendle S, Freudenthaler C, Gantner Z, Schmidt-Thieme L (2012) Bpr: Bayesian personalized ranking from implicit feedback. [arXiv:1205.2618](https://arxiv.org/abs/1205.2618)
52. Li Z, Ji J, Fu Z, Ge Y, Xu S, Chen C, Zhang Y (2021) Efficient non-sampling knowledge graph embedding. In: Proceedings of the web conference 2021, pp 1727–1736
53. Wu C, Wu F, Huang Y (2021) Rethinking infonce: How many negative samples do you need? [arXiv:2105.13003](https://arxiv.org/abs/2105.13003)
54. Mandal S, Maiti A (2022) Heterogeneous trust-based social recommendation via reliable and informative motif-based attention. In: 2022 international joint conference on neural networks (IJCNN), IEEE, pp 1–8
55. Mandal S, Maiti A (2022) Network promoter score (neps): an indicator of product sales in e-commerce retailing sector. *Electron Mark* 32(3):1327–1349
56. Abbasi-Moud Z, Vahdat-Nejad H, Sadri J (2021) Tourism recommendation system based on semantic clustering and sentiment analysis. *Expert Syst Appl* 167:114324
57. Singh DKS, Nithya N, Rahunathan L, Sanghavi P, Vaghela RS, Manoharan P, Hamdi M, Tunze GB (2022) Social network analysis for precise friend suggestion for twitter by associating multiple networks using ml. *Int J Inform Technol Web Eng* 17(1):1–11
58. Yu J, Gao M, Yin H, Li J, Gao C, Wang Q (2019) Generating reliable friends via adversarial training to improve social recommendation. In: 2019 IEEE international conference on data mining (ICDM), IEEE, pp 768–777
59. Lu Z, Gao M, Wang X, Zhang J, Ali H, Xiong Q (2019) Srrl: select reliable friends for social recommendation with reinforcement learning. In: Neural information processing: 26th international conference, ICONIP 2019, Sydney, NSW, Australia, December 12–15, 2019, Proceedings, Part II 26. Springer, pp 631–642