LighterKGCN: A Recommender System Model Based on Bi-layer Graph Convolutional Networks

Peng Chen, Jiancheng Zhao, Xiaosheng Yu*

College of Computer and Information Technology, China Three Gorges University, China
chenpeng@ctgu.edu.cn, 994106602@qq.com, yuxiaosheng@ctgu.edu.cn

Abstract

Recommender systems have been extensively utilized to meet users' personalized needs. Collaborative filtering is one of the most classic algorithms in the recommendation field. However, it has problems such as cold start and data sparsity. In that case, knowledge graphs and graph convolutional networks have been introduced by scholars into recommender systems to solve the above problems. However, the current graph convolutional networks fail to give full play to the advantages of graph convolution since they are employed either in the embedding representations of users and commodity entities, or in the embedding representations between entities of the knowledge graphs. Therefore, LighterKGCN, a recommender system model based on bi-layer graph convolutional networks was proposed in accordance with the KGCN model and the LightGCN model. In the first layer of GCN, the model first learned the embedding representations of users and commodity entities on the user-commodity entity interaction graph. Then, the attained user embedding and commodity embedding were used as the data source for the second layer of GCN. In the second layer, the entity v and its neighborhoods were calculated using the hybrid aggregation function proposed in this paper. The result was taken as the new entity v. According to tests on three public datasets and comparison results with the KGCN, LighterKGCN improved by 0.52% and 51.16% in terms of AUC and F1 performances, respectively on the dataset of MovieLens-20M; LighterKGCN improved by 0.67% and 45.0% in terms of AUC and F1 performances, respectively on the dataset of Yelp2018; and the number was 0.67% and 36.35% in AUC and F1 performances, respectively on the dataset of Last.FM.

Keywords: LighterKGCN, Embedding, Recommender system, Collaborative filtering, Knowledge graph

1 Introduction

In order to meet users' personalized needs, recommender systems have been extensively applied in fields such as social media, e-commerce, and news recommendation [1]. Its kernel is to predict commodities that users are most likely to click or purchase next time and show these commodities to users according to historical interactions such as user purchases and clicks.

Collaborative filtering (CF), one of the most classic recommendation algorithms in recommender systems, can predict commodities based on embedding representations of users and commodities obtained through the user-commodity interaction graph [2-7]. However, it faces problems such as cold start and data sparsity. In early stages, user ID and product ID were directly mapped into embedding representations in matrix factorization (MF) [8]. SVD is a classic content-based model which is recommended by potential (hidden) factors [9]. LibFM is a feature-based decomposition model in the CTR scenario [10]. The model connects user identification with commodity identification as the inputs of LibFM. LibFM is extended by appending the entity representations learned by TerE to each pair of user and commodity in LibFM +TerE [11]. Afterwards, the MF interaction function was replaced by a nonlinear neural network in the neural collaborative filtering model [4]. Recently, knowledge graphs (KG) or graph convolutional networks (GCN) have been introduced into the recommender systems to address cold start and enhance model performance. Besides, KGCN samples the neighbors of each entity of KG as its receptive field prior to calculating the representations of given entities by combining the neighbor information with the deviation so that the high-order structural information and semantic information of KG can be automatically discovered [12]. KGAT is oriented at embedding attention into the propagation layer that can adaptively propagate the embeddings from node neighbors to update node representations [13]. Inspired by graph convolutional networks [14-15], NGCF follows the same propagation rules as GCN, namely, further clarifying the embedding representations of entities through feature conversion, neighborhood aggregation, and nonlinear activation [6]. Moreover, LightGCN enhances the model performance by removing feature conversion and nonlinear activation based on NCGF [16].

The above methods significantly improve recommendation performance, especially the recommender systems adopting KG and GCN. However, they apply CGN either in the embedding representations of users and commodities, or in the embedding representations between entities in KG, failing to take full advantage of CGN's capability of extracting feature information by combining KG and CGN.

To address the above problems, LighterKGCN, a recommender system model based on bi-layer GCN (KGCN model and the LightGCN model), was proposed in the paper. The first layer of GCN was utilized to learn the embedding

*Corresponding Author: Xiaosheng Yu; E-mail: yuxiaosheng@ctgu.edu.cn
DOI: 10.53106/160792642022052303020

representations of users and commodity entities on the user-commodity entity interaction graph. Then, the acquired user embeddings and commodity embeddings were deemed as the data source for the second layer of GCN. In the second GCN layer, the entity v and its neighborhoods were calculated using the proposed hybrid aggregation function, with the result as the new entity v. According to the tests on two public datasets (MovieLens-20M and Last.FM), the LighterKGCN model is superior to other latest benchmark models. Compared with KGCN, LighterKGCN enhanced the performances of AUC and F1 by 0.52% and 51.16% on the dataset MovieLens-20M; compared with KGCN, LighterKGCN improved the performances of AUC and F1 by 0.67% and 45.0% on the dataset Yelp2018. compared with KGCN, LighterKGCN improved the performances of AUC and F1 by 0.48% and 36.35% on the dataset Last.FM.

The author's contributions in this paper are summarized as follows:

LighterKGCN was proposed. The first layer of GCN was adopted to learn embedding representations of users and commodity entities, and the above embedding representations were considered the data source of the second GCN layer.

Based on three commonly-used aggregate functions, a hybrid aggregate function was obtained.

The model was proved to perform better than the most advanced comparison model on three real-world data sets.

2 Related Work

2.1 LightGCN

2.1.1 Fundamental Principles

Light Graph Convolution Network (LightGCN) model, as illustrated in Figure 1. In LGC, only the normalized sum of neighbor embeddings is performed towards the next layer; other operations like self-connection, feature transformation, and nonlinear activation are all removed, which largely simplifies GCNs. In Layer Combination, sum over the embeddings at each layer to obtain the final representations. LightGCN is designed for simplifying GCN, making it more concise and suitable for recommender systems [16]. Based on NGCF, LightGCN eliminates feature conversion and nonlinear activation, two common designs in GCN. Experimental results prove that feature conversion and nonlinear activation can increase training difficulty without improving the CF performance. Hence, neighborhood aggregation is retained in LightGCN for CF. Finally, the experiments show that LightGCN performs better than NGCF. The propagation rule in LightGCN is defined as [6]:

$$\mathrm{e}_u^{(k+1)} = \sum_{i \in N_u} \frac{1}{\sqrt{|N_u|}\sqrt{|N_i|}} e_i^{(k)} \tag{1}$$

$$\mathrm{e}_i^{(k+1)} = \sum_{u \in N_i} \frac{1}{\sqrt{|N_i|}\sqrt{|N_u|}} e_u^{(k)} \tag{2}$$

Where, N_u is the number of neighbors of the user u; N_i is the number of neighbors of the commodity i; and $\frac{1}{\sqrt{|N_u|}\sqrt{|N_i|}}$ is symmetrical normalization in line with the design of the standard GCN [15], avoiding the case in which the scale of embeddings increases with the graph convolution operation [16].

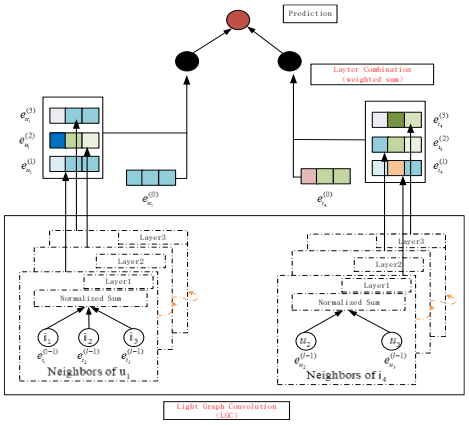

Figure 1. An illustration of LightGCNmodel architecture [16]

Assuming that the embedding propagation layer has a total of L layers, the final embedding representations of users and commodities are presented in (3):

$$e_u = \sum_{k=0}^{L} \alpha_k e_u^{(k)}; \quad e_i = \sum_{k=0}^{L} \alpha_k e_i^{(k)} \tag{3}$$

Where, α_k represents the weight of the kth layer during the formation of the final embedding. The experiments show that the best performance can be found when α_k is unified as $1/L+1$.

Finally, the model prediction results are served as the final recommendation and defined as:

$$\hat{\mathrm{y}}_{ui} = e_u^T e_i \tag{4}$$

2.1.2 Matrix Representation

Let $R \in R^{M \times N}$ be the interaction matrix between users and commodity entities in KG, where M is the number of users; and N is the number of commodities. If user u interacts with the commodity i, then $R_{ui} = 1$, otherwise $R_{ui} = 0$. On this basis, the adjacency matrix of the user's commodity graph can be obtained as:

$$A = \begin{pmatrix} 0 & R \\ R^T & 0 \end{pmatrix} \tag{5}$$

Let the 0th layer of the embedding matrix be $E^{(0)} \in R^{(M+N)\times T}$, where T is the embedding size, and therefore the equivalent form of the first GCN layer matrix can be expressed as:

$$E^{(k+1)} = (D^{-\frac{1}{2}} A D^{-\frac{1}{2}}) E^{(k)} \quad \textbf{(6)}$$

where, D is a $(M+N)\times(M+N)$ diagonal matrix, where D_{ii} stands for the number of non-zero records in the ith-row vector of the adjacency matrix A.

The final embedding matrix applied for model prediction can be expressed as:

$$E = \alpha_0 E^{(0)} + \alpha_1 E^{(1)} + \alpha_2 E^{(2)} + \ldots + \alpha_K E^{(K)} \quad \textbf{(7)}$$

It can also be expressed as:

$$E = \alpha_0 E^{(0)} + \alpha_1 \tilde{A} E^{(0)} + \alpha_2 \tilde{A}^2 E^{(0)} + \ldots + \alpha_K \tilde{A}^K E^{(0)} \quad \textbf{(8)}$$

Where, $\tilde{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ is the symmetric normalized matrix.

2.2 KGCN

To ease the sparsity and cold start of the recommender systems based on CF, Wang Hongwei et al. proposed KGCN and effectively captured the correlations between entities by mining their association attributes on the KG [12]. Moreover, KGCN sampled the neighbors of all entities on KG as their receptive fields and then combined the neighbor information with the deviation in calculating the representation of a given entity to automatically detect the high-order structural information and semantic information of KG. Note that the receptive field could be extended beyond multiple hops to simulate high-order adjacent information and capture users' potential long-distance interest. Moreover, in the above paper, the proposed KGCN was implemented in a small batch. In this way, the KGCN model could be operated on larger datasets and more complicated KG.

An example of two layers of receptive fields (green entities) of blue entities in KG with k=2 is presented in Figure 2. An iterative process of the KGCN algorithm is shown in Figure 3, where, the entity of a given node is represented by $e^u[h]$ and the neighbor representation $e_i^u[h]$ (green nodes) is mixed to form the next-iterative representation $e^u[h+1]$ (blue nodes).

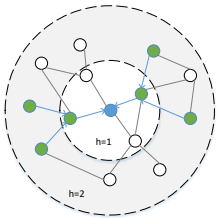

Figure 2. An example of two layers of receptive fields between entities, K=2 [12]

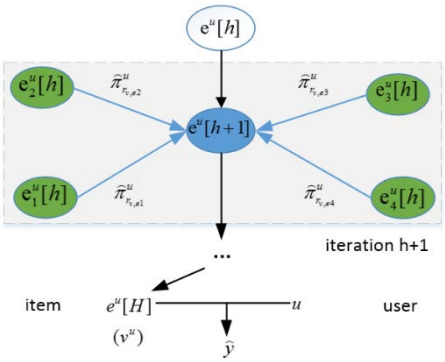

Figure 3. An iteration of the KGCN algorithm [12]

3 LighterKGCN

LighterKGCN is a recommendation system model based on a two-layer graph convolutional network. It is divided into two layers. In the first layer of GCN, the embedded representation of users and commodity entities is learned on the user-commodity entity interaction diagram, and then the obtained user and commodity embeddings are used as the data source for the second layer of GCN. In the second layer of GCN, the entity and its neighborhood are calculated using the hybrid aggregation function proposed in this paper, and the calculated result is used as a new entity. Finally, the embedded representation of users and commodity entities obtained by the second layer of GCN is used to calculate the predicted matching score.

3.1 A The First Layer of GCN

3.1.1 Aggregation Propagation Rule

The first GCN layer is designed to calculate embedding representations of users and commodities in KG. Feature conversion which is most commonly-used in GCN and nonlinear activation functions used in aggregation functions are removed from this layer. A new type of propagation rule adopted by the first GCN layer in this paper can be defined as:

$$e_u^{(k+1)} = \sum_{i \in N_u} e_i^{(k)} \quad \textbf{(9)}$$

$$e_i^{(k+1)} = \sum_{u \in N_i} e_u^{(k)} \quad \textbf{(10)}$$

What differs the propagation rules and the most adopted in the first GCN layer and LightGCN is the elimination of symmetric normalization. Furthermore, the function of adjusting the node loss rate is added in the model to facilitate model training and avoid over-fitting.

3.1.2 Hierarchical Embedding Representation

The higher-level embedding representations of users and commodities can be calculated via (9) and (10). After L-layer calculation, weighted summation is performed by (3) on the embeddings of various layers to obtain the end user embedding representation e_u and the commodity

embedding representation e_i upon the calculation of the first GCN layer.

3.1.3 Matrix Form

Let $R \in R^{M \times N}$ be the interaction matrix between users and commodity entities in KG. Where, M is the number of users; and N is the number of entities in KG. If the user u interacts with the commodity i, then $R_{ui} = 1$, otherwise $R_{ui} = 0$. Based on this, the adjacency matrix of the commodity graph of the user can be obtained as:

$$A = \begin{pmatrix} 0 & R \\ R^T & 0 \end{pmatrix} \tag{11}$$

Let the embedding matrix in the 0th layer be $E^{(0)} \in R^{(M+N) \times T}$, where T is the embedding size, and thus the equivalent form of the first GCN layer matrix can be expressed as:

$$E^{(k+1)} = AE^{(k)} \tag{12}$$

At last, the final embedding matrix of the first GCN layer after the L-layer calculation is obtained as:

$$E = \alpha_0 E^{(0)} + \alpha_1 E^{(1)} + \alpha_2 E^{(2)} + ... + \alpha_L E^{(L)} \tag{13}$$

It can also be expressed as:

$$E = \alpha_0 E^{(0)} + \alpha_1 AE^{(0)} + \alpha_2 A^2 E^{(0)} + ... + \alpha_L A^L E^{(0)} \tag{14}$$

3.2 The Second Layer of GCN

The KGCN model is the essence of the second GCN layer. Specifically, the initialization matrix of the original trainable user U and the initialization matrix of the trainable entity E are replaced by the embedding matrices of the user U and the entity E trained by the first GCN layer, respectively.

u and v are adopted to represent users and commodity entities, respectively. $N(\mathrm{v})$ stands for the collection of entities directly connected with v; and r indicates the entity relationship. The function g(●) is adopted to calculate the score between the users and the entity relationship:

$$\pi_r^u = g(u, r) = u \odot r \tag{15}$$

The neighborhood aggregation of the entity v is expressed as:

$$v_{N(v)}^u = \sum_{e \in N(v)} \tilde{\pi}_{r_{v,e}}^u e \tag{16}$$

Where, $\tilde{\pi}_{r_{v,e}}^u$ is the standardized result of $\pi_{r_{v,e}}^u$, $\pi_{r_{v,e}}^u$ indicates the score between the entity v and the neighbor e under the relationship r for the user u. Based on this, As shown in (17):

$$\tilde{\pi}_{r_{v,e}}^u = \frac{\exp(\pi_{r_{v,e}}^u)}{\sum_{e \in N(v)} \exp(\pi_{r_{v,e}}^u)} \tag{17}$$

Since the number of neighbors of the commodity v is uncertain, the neighboring representation is changed from $v_{N(v)}^u$ to $v_{S(v)}^u$ to ensure the simplicity and feasibility of the model, where, $|S(v)| = K$ and K are configurable constants for describing K neighbors that capture v.

After that, the entity v and its neighborhood representation $v_{S(v)}^u$ are aggregated into a new entity v. In other words, a new aggregation method is created in addition to KGCN aggregation to obtain a new entity v through aggregation. The new aggregation is expressed in (18) as below:

$$v = \alpha \cdot agg_{\mathrm{sum}}(v, v_{S(v)}^u) + \beta \cdot agg_{concat}(v, v_{S(v)}^u) + \gamma \cdot agg_{neighbor}(v, v_{S(v)}^u) \tag{18}$$

Where, α, β, and γ are trainable weight coefficients. The calculation result of the new entity v is derived from three aggregation methods. In this paper, (18) was named as a hybrid aggregation function. Agg is an aggregate function with three settings:

- Summator: Two representation vectors are summed before performing a nonlinear transformation:

$$agg_{sum} = \sigma(W \cdot (v + v_{S(v)}^u) + b) \tag{19}$$

Where, W and b are the transformation weight and offset, respectively, and σ is the activation function.

- Connection aggregator: Two representation vectors are firstly connected before the nonlinear transformation [14]:

$$agg_{concat} = \sigma(W \cdot concat(v, v_{S(v)}^u) + b) \tag{20}$$

- Neighbor aggregator: The neighbor representation of the entity v is regarded as the output representation [17]:

$$agg_{neighbor} = \sigma(W \cdot v_{S(v)}^u + b) \tag{21}$$

The obtained entity v is considered as the input of the next training. Embedding representations of the entity v and user u are finally obtained upon H iterations, which are then introduced into the function $f(\cdot)$ upon L2 regularization processing for probability prediction:

$$\hat{y}_{uv} = f(u, v) = u \odot v \tag{22}$$

Where, $\hat{y}_{uv}$ represents the probability of the user u interacting with the commodity v. Precisely, when a model is greater than or equal to 0.5, the user u will be considered to interact with the commodity v, otherwise no interaction is considered between the user u and the commodity v.

At last, a negative sampling strategy is adopted in training to enhance the calculation efficiency. At the same time, the loss function set in KGCN is maintained [12], as shown in (23):

$$\ell = \sum_{u \in U} \Big(\sum_{v: y_{uv}=1} \Im(y_{uv}, \hat{y}_{uv}) - \sum_{i=1}^{T^u} E_{v_i \square P(v_i)} \Im(y_{uv_i}, \hat{y}_{uv_i}) \Big) + \lambda \| F \|_2^2 \quad (23)$$

Where, $\Im$ is the loss of cross entropy; P is the negative sampling distribution; T^u is the negative sampling number of the user u; $T^u = \left| \{ v : y_{uv} = 1 \} \right|$ and P are in line with homogeneous distribution. $\| F \|_2^2$ represents L2 regularization.

4 Experiment

In this paper, three real-world datasets were experimented in order to evaluate the proposed method and answer the following three questions.

Q1: How does LighterKGCN perform in comparison with the existing methods?

Q2: Is the model affected by the number of different aggregation layers in the first layer of GCN?

Q3: How is the performance of the proposed hybrid aggregate function compared with other aggregate functions?

4.1 Dataset

To assess the effectiveness of LighterKGCN, MovieLens-20M, Yelp2018 and Last.FM, two public datasets, were experimented with. Statistics of the two datasets is summarized in Table 1.

MovieLens-20M[1] This is a benchmark dataset widely used in movie recommendations, comprising of approximately 20 million scores ranging from 1 to 5 on the MovieLens website.

Yelp2018[2] This dataset is adopted from the 2018 edition of the Yelp challenge. Here we view the local businesses like restaurants and bars as the items.

Last.FM[3] This is the music listening dataset collected from Last.FM online music systems. Wherein, the tracks are viewed as the items.

MovieLens-20M, as explicit feedback, should be converted into implicit feedback in the experiment. Therefore, the datasets provided by KGCN are implicit feedback in this paper [12]. In order to ensure the consistency of the results processed by the dataset Yelp2018 and Last.FM, the dataset provided in KGAT is used [13]. About Yelp2018 and Last.FM dataset, X. Wang et al. strongly suggested to use the trained user and item embeddings of BPR-MF to initialize the user and item embeddings of all models [13]. Therefore, all codes in this paper first use BPR-MF trained user and project embedding to initialize models. The code implementation of the comparison method uses the code provided in KGAT [13]. Because our method and KGCN do not explicitly specify the division into training set and test set in the code implementation, this paper divides the item interacted by users into training set and test set in the form of 8:2. In addition, the evaluation method is also implemented according to the original code.

Table 1. Statistics of datasets

	Movie	Yelp2018	Last.FM
#users	138,159	45,919	23,566
#items	16,954	45,538	48,123
#interactions	13,501,622	1,185,068	3,034,796
#entities	102,569	90,961	58,266
#relations	32	42	9
#KG triples	499,474	1,853,704	464,567

4.2 Comparison Method

To prove the model effectiveness, LighterKGCN is compared with the following methods:

- CKE adopt a heterogeneous network embedding method [18], termed as TransR [19], to extract items' structural representations by considering the heterogeneity of both nodes and relationships.
- CFKG propose a knowledge-base representation learning framework to embed heterogeneous entities for recommendation [20]. The model applies TransE [11] on the unified graph including users, items, entities, and relations.
- KGAT explicitly models the high-order connectivities in KG in an end-to-end fashion [13]. It recursively propagates the embeddings from a node's neighbors (which can be users, items, or attributes) to refine the node's embedding, and employs an attention mechanism to discriminate the importance of the neighbors.
- The RippleNet model enriches their representations with a multi-hop path rooted at each user in KG and makes predictions on the representations using MF [21].
- KGCN can capture high-order structures and semantic information in KG automatically with the key logic of gathering and merging neighborhood information with deviations in the process of calculating the representations of a given entity in KG [12].

4.3 Parameter Setting

The node_dropout hyper-parameter was added in LighterKGCN to set the node dropout rate within the range between 0.0 and 1.0. In addition, the node_dropout_flag hyper-parameter was utilized to decide whether node dropout is enabled. The mat hyper-parameter was used to set the aggregation mode of the first GCN layer within the integer range between 0 and 3. To sum up, four aggregation modes are available, namely, not normalized, the left side is normalized after adding the identity matrix, the normalization on the left,the left normalization and the right normalization. The n_layers hyper-parameter is for setting the number of aggregation layers of the first GCN layer. H represents the number of aggregation layers of the second GCN layer. The aggregator represents the aggregation mode of the second GCN layer, including sum, concat, neighbor, and mix. To be specific, mix is the proposed hybrid aggregation method,

[1] https://grouplens.org/datasets/movielens/

[2] https://www.yelp.com/dataset/challenge

[3] https://grouplens.org/datasets/hetrec-2011/

which is presented in (18). Specific parameter settings of LighterKGCN are shown in Table 2.

Table 2. Parameter setting of LighterKGCN

	Movie	Yelp2018	Last.FM
K	4	8	8
d	64	64	64
H	3	1	1
λ	10^{-7}	2×10^{-6}	10^{-6}
η	5×10^{-4}	5×10^{-5}	5×10^{-5}
batch size	16384	1024	1024
aggregator	mix	mix	mix
node_dropout	0.0	0.0	0.0
mat	0	3	3
n_layers	1	1	1

Note: K: neighbor sampling size, d: embedding dimension, λ: L2 regularized weight, η: learning rate.

Moreover, AUC and F1 were also taken as the evaluation indicators in the experiment like the evaluation method in KGCN. The formula for calculating the F1 value is shown in formula (26), where formula (24) is the precision rate, and formula (25) is the recall rate. The value of AUC is the area under the ROC curve, the ordinate of the ROC curve is the "True Case Rate" (TFR), and the abscissa is the "False Positive Rate" (FPR). From the definition, it can be seen that AUC can be obtained by summing the area of each part under the ROC curve. Assuming that the ROC curve is formed by connecting points with coordinates $\{(x_1,y_1)(x_2,y_2)...(x_m,y_m)\}$ in a sequence (x_1=0, x_m=1), the AUC can be estimated as the formula (27) Shown. In the experiment, in order to evaluate the effectiveness of top-K recommendation and to facilitate comparison with other methods, we set K to 20 when calculating the F1 value, that is, each user calculates the item with the highest recommended value. The first 20 items corresponding to the test set are calculated to obtain the F1 value.

$$\text{precision}=\frac{TP}{TP+FP} \tag{24}$$

$$\text{recall}=\frac{TP}{TP+FN} \tag{25}$$

$$F1=\frac{2\times precision\times recall}{precision+recall} \tag{26}$$

$$AUC=\frac{1}{2}\sum_{i=1}^{m-1}(x_{i+1}-x_i)\cdot(y_i+y_{i+1}) \tag{27}$$

Where, TP is a true positive, FP is a false positive, and FN is a false negation.

Table 3. Comparison between LighterKGCN and other methods

Model	Movie		Yelp2018		Last.FM	
	AUC	F1	AUC	F1	AUC	F1
CKE	0.9770	0.1611	0.9502	0.0216	0.9157	0.0433
CFKG	0.9704	0.1227	0.9434	0.0206	0.8836	0.0419
KGAT	0.9782	0.1809	0.9625	0.0262	0.9514	0.0626
RippleNet	0.9729	0.0983	0.9571	0.0254	0.9420	0.0557
KGCN	0.9788	0.1210	0.9694	0.0380	0.9645	0.0861
LighterKGCN	**0.9839**	**0.1829**	**0.9759**	**0.0551**	**0.9692**	**0.1174**
%Improv.	0.52%	51.16%	0.67%	45.0%	0.48%	36.35%

Note: %Improv. is calculated based on KGCN

4.4 Result

4.4.1 Performance Comparison

The performance comparison results in Table 3 show the following points:

- KGCN has almost the best performance in the AUC and F1 scores of both the MovieLens-20M dataset, Yelp2018 dataset and Last.FM dataset. KGCN extends the non-spectral GCN method to the KG and gathers neighborhood information in a selective and biased manner, which can not only learn the structural information and semantic information of KG, but also learn the users' personalized needs and potential interests, thus being conducive to performance improvement.
- It can be found that the performance of KGAT on the AUC and F1 scores is better than that of CKE, CFKG and RippleNet. This is because KGAT can explore high-level connectivity in an explicit way, thereby effectively capturing cooperative signals. This verifies the importance of capturing knowledge of collaborative signaling.
- LighterKGCN always produces the best performance in MovieLens-20M dataset, Yelp2018 dataset and Last.FM dataset. On the MovieLens-20M dataset, compared with KGCN, AUC and F1 have improved by 0.52% and 51.16%, respectively. On the Yelp2018 dataset, compared with KGCN, AUC and F1 have improved by 0.67% and 45.0%, respectively. On the Last.FM dataset, compared with KGCN, AUC and F1 have increased by 0.48% and 36.35%, respectively. This also verifies that by adding a layer of GCN to calculate the embedded representation of users and commodity entities, the performance of the model can be significantly improved.
- At the same time, it can also be found that in the performance of the above three datasets, LighterKGCN's AUC has improved very little, but it has improved a lot in F1. The possible reason is that the AUC value of KGCN in these three datasets is

already very high, even if the LighterKGCN is used, the AUC improvement is very limited.

4.4.2 Influence of Different Layers on the First Layer of GCN

The model depth is changed, and the range of layers is set {1,2,3} to examine whether the first round of GCN of LighterKGCN benefits from multiple layers of embedding propagation. Experimental results are summarized in Table 4. LighterKGCN-3 indicates a model with three embedding propagation layers. It is worth noting that similar symbols can also be found in other models. The "-" in the table shows that a given model is omitted due to its poor performance and excessively long training time. From the analysis in Table 4, the following results can be obtained:

- In the three datasets, the lower the number of embedding propagation layers, the better the performance. The reason for this phenomenon is that the data in the data can be embedded and propagated at only one layer to obtain a wealth of information. If the number of layers of embedding and propagation is further increased, more useless information will be obtained.
- It can also be found from the above table that the more data in the dataset, the more obvious the performance degradation of increasing the number of embedding propagation layers, especially on the MovieLens-20M dataset, followed by Yelp2018, and finally Last.FM.

On the MovieLens-20M dataset, in the presence of over two embedding propagation layers, the model performance will be impaired dramatically accompanied by a huge increase in training time. Evidently, multiple propagation layers don't necessarily enhance model performance. Hence, the appropriate number of layers should be selected as per the actual situation.

Table 4. Influence of the number of layers of embedding propagation

Model	Movie		Yelp2018		Last.FM	
	AUC	F1	AUC	F1	AUC	F1
LighterKGCN-1	**0.9839**	**0.1829**	**0.9725**	**0.0551**	**0.9692**	**0.1174**
LighterKGCN-2	0.9671	0.1752	0.9688	0.0431	0.9641	0.0961
LighterKGCN-3	-	-	0.9663	0.3762	0.9590	0.0902

4.4.3 Comparison of Different Aggregation Methods in the First Layer of GCN

The following points can be concluded from the test results of all the aggregate functions shown in Table 5:

Table 5. Influence of varied aggregate functions

Model	Movie		Yelp2018		Last.FM	
	AUC	F1	AUC	F1	AUC	F1
$\text{LighterKGCN}_{\text{sum}}$	0.9822	0.1779	0.9641	0.0365	0.9618	0.0655
$\text{LighterKGCN}_{\text{concat}}$	0.9826	0.1807	0.9695	0.0383	0.9496	0.0673
$\text{LighterKGCN}_{\text{neighbor}}$	0.9775	0.1424	0.8713	0.0175	0.9160	0.0221
$\text{LighterKGCN}_{\text{mix}}$	**0.9839**	**0.1829**	**0.9725**	**0.0551**	**0.9692**	**0.1174**

- The proposed hybrid aggregate function produces the best model performance.
- The concat aggregate function has the smallest gap with the mix aggregate function in terms of performance.
- No matter which dataset it is in, the performance of neighbor aggregation function is the worst.

4.4.4 Hypothesis Testing

In order to compare the advantages of LighterKGCN and other comparison methods in AUC and F1 more comprehensively, hypothesis testing is used here for testing.

The null hypothesis about the AUC value is: H0: In 3 different experimental datasets, there is no difference in AUC between LighterKGCN and the popular comparison method. In order to reject this hypothesis, the Friedman rank sum test was used in this experiment to test the significant difference between multiple methods. First, the Friedman rank sum test ranks the AUC values, where the highest F1 value is assigned to the first level, the second highest F1 value is assigned to the second level, and so on. Finally, Friedman's test compares the average ranks of the methods. Table 6 shows the AUC grades of LighterKGCN and popular comparison methods and their average grades in the 3 datasets.

Table 6. The AUC level of LighterKGCN and the popular comparison method and the average level of the 3 datasets

Dataset	CKE	CFKG	KGAT	RippleNet	KGCN	LighterKGCN
MovieLens-20M	4	6	3	5	2	1
Yelp2018	5	6	3	4	2	1
Last.FM	5	6	3	4	2	1
AVG	4.66	6	3	4.33	2	1

Note: AVG is the average.

If the experimental result satisfies the null hypothesis, it means that the execution of all algorithms is similar, so their average rank R_j should be equal. For the calculation of Friedman statistics, please refer to formula (28):

$$\chi_F^2 = \frac{12 \cdot N}{k \cdot (k+1)} [\sum_j R_j^2 - \frac{k \cdot (k+1)^2}{4}] \quad \textbf{(28)}$$

Because Iman and Davenport [22] claimed that Friedman's was too conservative, they introduced better new statistics, see formula (29):

$$F_F = \frac{(N-1) \cdot \chi_F^2}{N \cdot (\mathrm{k}-1) - \chi_F^2} \quad \textbf{(29)}$$

The metric is assigned according to the F distribution with k-1 and $(k-1) \cdot (N-1)$ degrees of freedom. If the negative hypothesis is rejected, a post-test is required to discover the key difference between the average levels of these models.

This article uses the 95% confidence interval ($\alpha = 0.05$) as the threshold for rejecting the null hypothesis, and uses Friedman's test to calculate the F distribution:

$$\chi_F^2 = \frac{12 \times 3}{6 \times (6+1)} [(4.66^2 + 6^2 + 3^2 + 4.33^2 + 2^2 + 1^2) - \frac{6 \times (6+1)^2}{4}] = 14.541$$

$$F_F = \frac{(3-1) \times \chi_F^2}{3 \times (6-1) - \chi_F^2} = \frac{2 \times 14.541}{15 - 14.541} = 63.359$$

This experiment has 3 datasets and 6 comparison methods. According to the F_F distribution, it has 6-1=5 and (6-1)·(3-1)=10 degrees of freedom, the critical value $\alpha = 0.05$ of the significance level of $F(2,10)$ is 4.354. $F_F > F(2,10)$ is observed, so the null hypothesis is rejected. This also means that there is a difference in the performance of AUC between LighterKGCN and the popular comparison method on the considered dataset.

The null hypothesis about the F1 value is: H0: In 3 different experimental datasets, there is no difference between LighterKGCN and the popular comparison method in F1. In order to reject this hypothesis, the Friedman rank sum test was used in this experiment to test the significant difference between multiple methods. First, the Friedman rank sum test ranks F1 values, where the highest F1 value is assigned to the first level, the second highest F1 value is assigned to the second level, and so on. Finally, Friedman's test compares the average ranks of the methods. Table 7 shows the F1 grades of LighterKGCN and the popular comparison method and their average grades in the 3 datasets.

Table 7. The F1 level of LighterKGCN and the popular comparison method and the average level of the 3 datasets

Dataset	CKE	CFKG	KGAT	RippleNet	KGCN	LighterKGCN
MovieLens-20M	3	4	2	6	5	1
Yelp2018	5	6	3	4	2	1
Last.FM	5	6	3	4	2	1
AVG	4.33	5.33	2.66	4.66	3	1

Note: AVG is the average.

$$\chi_F^2 = \frac{12 \times 3}{6 \times (6+1)} [(4.33^2 + 5.33^2 + 2.66^2 + 4.66^2 + 3^2 + 1^2) - \frac{6 \times (6+1)^2}{4}] = 10.671$$

$$F_F = \frac{(3-1) \times \chi_F^2}{3 \times (6-1) - \chi_F^2} = \frac{2 \times 10.671}{15 - 10.671} = 4.930$$

The critical value $\alpha = 0.05$ of the significance level of $F(2,10)$ is 4.354. $F_F > F(2,10)$ is observed, so the null hypothesis is rejected. This also means that there is a difference in the performance of F1 between LighterKGCN and the popular comparison method on the considered dataset.

5 Conclusion

In this paper, LighterKGCN, a recommender system model based on bi-layer graph convolutional networks was proposed in accordance with the KGCN model and the LightGCN model. The first layer of GCN learned embedding representations of user-commodity entities on the user-commodity entity interaction graph. Next, the obtained user embedding and commodity embedding representations were used as the data source for the second layer of GCN. In the meantime, the proposed hybrid aggregate function was adopted as the aggregate function of the second layer of GCN. In the end, various experiments were performed on three real-world datasets, proving that it is effective and feasible to learn the embedding representations between users and entities in knowledge graphs by adding a layer of GCN besides KGCN.

In future work, the attention mechanism will be introduced in the first layer of GCN to enhance the model performance. In addition, more supporting information such as social networks will be used for recommendations [23].

References

[1] A. A. J. Jothi, R. A. Sulthana, A Review on the Literature of Fashion Recommender System using Deep Learning, *International Journal of Performability Engineering*, Vol. 17, No. 8, pp. 695-702, August, 2021.

[2] Z. Y. Cheng, Y. Ding, L. Zhu, M. Kankanhalli, Aspect-Aware Latent Factor Model: Rating Prediction with Ratings and Reviews, *The International Conference of World Wide Web*, Lyon, France, 2018, pp. 639-648.

[3] T. Ebesu, B. Shen, Y. Fang, Collaborative Memory Network for Recommendation Systems, *Special Interest Group on Information Retrieval*, Ann Arbor, Michigan, USA, 2018, pp. 515-524.

[4] X. N. He, L. Z. Liao, H. W. Zhang, L. Q. Nie, X. Hu, T.-S. Chua, Neural Collaborative Filtering, *The*

International Conference of World Wide Web, Perth, Australia, 2017, pp. 173-182.

[5] D. W. Liang, Rahul. G. Krishnan, D. M. Hoffman, T. Jebara, Variational Autoencoders for Collaborative Filtering, *The International Conference of World Wide Web*, Lyon, France, 2018, pp. 689-698.

[6] X. Wang, X. N. He, M. Wang, F. L. Feng, T.-S. Chua, Neural Graph Collaborative Filtering, *Special Interest Group on Information Retrieval*, Paris, France, 2019, pp. 165-174.

[7] C. Su, D. Huang, Hybrid Recommender System based on Deep Learning Model, *International Journal of Performability Engineering*, Vol. 16, No. 1, pp. 118-129, January, 2020.

[8] Y. Koren, R. Bell, C. Volinsky, Matrix Factorization Techniques for Recommender Systems, *IEEE Computer*, Vol. 42, No.8, pp. 30-37, August, 2009.

[9] Y. Koren, Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model, *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, Las Vegas, Nevada, USA, 2008, pp. 426-434.

[10] S. Rendle, Factorization Machines with Libfm, *ACM Transactions on Intelligent Systems and Technology (TIST)*, Vol. 3, No. 3, pp. 1-22, May, 2012.

[11] A. Bordes, N. Usunier, A. G. Duran, J. Weston, O. Yakhnenko, Translating Embeddings for Modeling Multi-relational Data, *Neural Information Processing Systems*, Lake Tahoe, Nevada, USA, 2013, pp. 2787-2795.

[12] H. Wang, M. Zhao, X. Xie, W. Li, M. Guo, Knowledge Graph Convolutional Networks for Recommender Systems, *The International Conference of World Wide Web*, San Francisco, CA, USA, 2019, pp. 3307-3313.

[13] X. Wang, X. N. He, Y. X. Cao, M. Liu, T.-S. Chua, KGAT: Knowledge Graph Attention Network for Recommendation, *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, Anchorage, Alaska, USA, 2019, pp. 950-958.

[14] W. L. Hamilton, R. Ying, J. Leskovec, Inductive Representation Learning on Large Graphs, *Neural Information Processing Systems*, Long Beach, CA, USA, 2017, pp. 1025-1035.

[15] T. N. Kipf and M. Welling, Semi-supervised Classification with Graph Convolutional Networks, *in Proc. International Conference on Learning Representations*, Toulon, France, 2017, pp. 24-26.

[16] X. N. He, K. Deng, X. Wang, Y. Li, Y. D. Zhang, M. Wang, LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation, *Special Interest Group on Information Retrieval*, Virtual Event, China, 2020, pp. 639-648.

[17] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, *International Conference on Learning Representations*, Vancouver, Canada, 2018, pp. 1-12.

[18] F. Z. Zhang, N. J. Yuan, D. Lian, X. Xie, W. Y. Ma, Collaborative Knowledge Base Embedding for Recommender Systems, *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 2016, pp. 353-362.

[19] S. Z. Dai, Y. C. Liang, S. Y. Liu, Y. Wang, W. L. Shao, X. X. Lin, X. Y. Feng, Learning Entity and Relation Embeddings with Entity Description for Knowledge Graph Completion, *Proceedings of 2018 2nd International Conference on Artificial Intelligence: Technologies and Applications (ICAITA2018)*, Chengdu, China, 2018, pp. 194-197.

[20] Q. Y. Ai, V. Azizi, X. Chen, Y. F. Zhang, Learning Heterogeneous Knowledge Base Embeddings for Explainable Recommendation, *Algorithms*, Vol. 11, No. 9, Article No. 137, September, 2018.

[21] H. W. Wang, F. Z. Zhang, J. L. Wang, M. Zhao, W. J. Li, X. Xie, M. Y. Guo, RippleNet: Propagating User Preferences on the Knowledge Graph for Recommender Systems, *ACM International Conference on Information and Knowledge Management*, Lingotto Turin, Italy, 2018, pp.417-426.

[22] R. L. Iman, J. M. Davenport, Approximations of the critical region of the fbietkan statistic, *Communications in Statistics- Theory & Methods*, Vol. 9, No. 6, pp. 571-595, September, 1980.

[23] L. Wu, P. J. Sun, Y. J. Fu, R. C. Hong, X. T. Wang, M. Wang, A Neural Influence Diffusion Model for Social Recommendation, *in Proc. Special Interest Group on Information Retrieval*, Paris, France, 2019, pp. 235-244.

Biographies

Peng Chen was born in Enshi, Hubei, China in 1973. He received the Ph.D. in system analysis and integration from the Huazhong University of Science and Technology. Now, he has been a Professor with Computer and Information Institute of China Three Gorges University. His research interests include Artificial Intelligence, Big Data.

Jiancheng Zhao received his B.S. degree in Information management and information system from Henan Polytechnic University, Henan, China, in 2018. Now he is studying for a master's degree at China Three Gorges University.

Xiaosheng Yu was born in Jianli, Hubei, China in 1973. He received the Ph.D. degree in information science from Wuhan University in 2007. Since 2010, he has been an associate professor with Computer and Information Institute of China Three Gorges University. His research interests include big data analysis, information fusion.