




Neighbor-Augmented Knowledge Graph Attention Network for Recommendation

Qi Wang^{1,2} · Hao Cui¹ · Jiapeng Zhang¹ · Yan Du¹ · Yuan Zhou¹ · Xiaojun Lu¹ 

Accepted: 22 May 2023

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract

Previous knowledge graph-based recommendation models use Graph Neural Networks to aggregate information from neighbors. However, entities of the same hop should have different importance in aggregating, but this is often ignored. To overcome this problem, we try to utilize more helpful information to enhance the recommender system. This paper presents a new recommendation model: Neural Attention Network for Neighborhood Augmented Knowledge Graph (NKGAT). The model is developed based on knowledge graph and graph neural network. In the proposed model, we explore higher-order relationships using embeddings of nodes. Moreover, a new attention module (Neighbor-Augmented Attention (NAA)) is proposed. It assigns proper importance to different entities of neighbors at the same hop in the graph neural network. The experiments are carried out on Movie-Lens, Last-FM, Book-Crossing and Yelp2018 datasets. The experimental results show that NKGAT achieves state-of-the-art performance. It demonstrates that the proposed method integrates rich higher-order information with more proper attention to entities.

Keywords Neighbor-augmented attention · Recommender system · Graph neural network · Knowledge graph · Collaborative filtering

✉ Xiaojun Lu
luxiaojun@mail.neu.edu.cn

Qi Wang
wangqimath@mail.neu.edu.cn

Hao Cui
cuihao30@jd.com

Jiapeng Zhang
1970034@stu.neu.edu.cn

Yan Du
dy13998268481@163.com

Yuan Zhou
zhouyuan01@ln.chinamobile.com

¹ College of Sciences, Northeastern University, Shenyang, 110819, China

² Key Laboratory of Data Analytics and Optimization for Smart Industry (Northeastern University), Ministry of Education, Shenyang, 110819, China

1 Introduction

How to accurately capture users' preferences through their historical behaviors? It is still a challenging problem for personalized recommendations.

Collaborative Filtering (CF) [1–4] is a classical model from the perspective of item similarity or user similarity. It assumes that users with similar interaction histories may have similar interests in items. But the co-occurrence matrix in CF is quite sparse and difficult to process. To enhance the generalization ability of the model, matrix factorization (MF) [5] is introduced to recommender system. However, the MF-based approaches ignore the higher-order connectivity relationships that exist in the user-item interaction graph. Sun et al. [6] propose a deep plot-aware generalized matrix decomposition for collaborative filtering models that effectively combines ratings and plot text to achieve generalized collaborative filtering. Factorization machine(FM) [7] utilizes the second-order cross feature, it can combine different features. Furthermore, the domain-aware factorization machine(FFM) [8], developed from FM, further enhances the ability of factorization machine feature cross-over by adding the concept of feature domain.

In recent years, methods based on Graph Neural Network(GNN) that incorporate deep learning have become popular. GNNs [9] approximate functions in the graph domain. It means that GNNs can process graph data in their native form. It can adapt the network architecture to the input native graph data. Moreover, GNNs are function approximators in the graph domain under some conditions [9]. GNNs adapt their architecture by replicating two MLPs over nodes and/or edges. More exactly, k-hop neighbor features can be utilized via k iterations of state updating. Deep learning models such as NGCF [3], LightGCN [10], and NIAGCN [11] treat users and items as bipartite graphs. It can propagate the interaction information between items and users through the graph's structure. This demonstrates that the high-level connections between users and items can improve the performance of collaborative filtering.

Knowledge Graph (KG) [12, 13] also plays an increasingly important role in recommender systems. It can address the sparsity and cold-start problems of collaborative filtering effectively. So KG is widely studied and adopted as auxiliary information in recommender systems. Compared with collaborative filtering, the recommendation model based on knowledge graphs can introduce more semantic information and discover users' interests accurately.

The feature-based approaches mainly extract user and item attributes from the knowledge graph. Then these features are combined with a traditional model. This method only inserts entity features and does not introduce relational features. Path-based approaches, such as FMG [14], consider the knowledge graph as a heterogeneous information network. They construct meta-paths or meta-graph-based features between items. In short, a meta-path is a specific path that connects two entities, for example, the meta-path "actor→film→director→film→actor" can connect two actors, so it can be seen as a way to explore potential relationships between actors. The advantage of this approach is that it fully and visibly utilizes the network structure of the knowledge graph, while the disadvantage is that the meta-path or meta-graph needs to be designed manually, which is difficult to achieve optimal in practice. Distance-based translation models use a distance-based scoring function to evaluate the probability of triples. These types of approaches are effective, such as TransE [15], TransH [16], TransR [17] etc. However, the relationship between neighbors belonging to the same hop has not been exploited, since the neighborhoods' information is helpful for recommendation, we try to combine this information to the recommender system.

The contribution of this paper is as follows:

- We combine KG and GNN to aggregate and propagate information about users and items by using different mapping matrices for entity mapping, which greatly optimizes the model's performance.
- We explore the relationship between neighbors which belong to the same hop in the knowledge graph structure, where similar items are extremely relevant to users' preferences, and we design an adaptive attention model named Neighbor-Augmented Attention (NAA) to collect more valid information.

2 Related Work

The existing recommendation methods combined with KG can be classified into four types generally.

Path-based Methods [18–22]: These methods predict users preferences by connecting the paths of target users and item nodes through entities. Methods such as RippleNet [23], store a representation of items with each user as the root. But these recommendation methods have some drawbacks. They depend on the quality of the paths. Such methods often use greedy algorithms to search. When the data constitutes a large-scale graph, it can cause a tremendous waste of resources.

Embedding-based Methods [24–29]: These methods use KG embeddings to embed user items into learning entities and then use them as preliminary information of the items to guide the model. For example, CKE [29] applies TransE [15] to KG triples to embed KG information of items into matrix decomposition. KTUP [25] uses TransH [16] on both user-item bipartite graphs and triples to learn users' preferences. All these methods validate the advantages of knowledge embedding. They ignore higher-order connectivity and fail to reflect long-term semantic information between two entities.

Policy-based Methods [30–35]: Reinforcement learning models find strategies by learning paths. For example, PGPR [32] uses policy networks to find the items of most interest to the target user. However, sparse rewards accompanied by huge action routes make it difficult to obtain satisfying solutions by policy gradient.

GNN-based Methods [36–41]: Based on the aggregation mechanism, the information of single-hop nodes is combined to update the expression of the current node. These methods recursively combine the information of multi-hop to enhance the feature expressing. For example, KGAT [38] combines user project interaction and KG into a heterogeneous graph. CKAN [39] uses two strategies to propagate signals separately. These approaches using GNN have good interpretation capabilities although they cannot preserve the information of the paths.

3 Proposed Method

3.1 Knowledge Graph

The TransE [15] module can model the relationship between entities. The model considers the triple $\langle h, r, t \rangle$ as a translation from the head entity h to the tail entity t using the relation r in the knowledge graph. In fact, the assumption of $h + r \approx t$ in the TransE [15] model is too strict. The vectors of entities in the self-inverse, pair-wise, and multi-pairwise relationships sometimes are learned improperly. Therefore, to solve this problem, the TransH

[16] model relaxes the strict assumption of $h + r \approx t$. It just requires that the projections of the head and tail entities on the hyperplane corresponding to the relation r are close to each other. The head entity vector h and the tail entity vector t are mapped on the hyperplane using the normal vector W_r as $h_{\perp} = h - W_r^T h W_r$ and $t_{\perp} = t - W_r^T t W_r$.

In TransE [15] and TransH [16], entities and relations are labeled in the same space. However, the methods may be incorrect for those cases whose the semantics of the entities are similar while the specific relations are different. The TransR [17] constructs a corresponding vector space for each relation to represent entities and relations separately in different vector spaces. It requires that the projections of head and tail entities are close in the vector space. But this introduces some new problems. Firstly, such an operation increases the computation. Secondly, for a triple, the head and tail entities are of different types, so it is not reasonable to share a common mapping matrix. The TransD [42] can effectively solve the problems of the above models. It assumes that the mapping function should relate to entities and relations. TransD [42] modifies the mapping function by two different mapping matrices for the head entity and the tail entity. Compared with TransR [17], TransD [42] is converted from matrix calculation to inter-vector operation, and the speed of operation is greatly improved. All of the above models can be used as Encoder, and we finally use the TransD [42] to get the optimal results. The models are illustrated in Fig. 2.

3.2 Higher-Order Connectivity

Figure 3 shows a higher-order connectivity diagram. Consider the following connectivity $u_2 \xrightarrow{r_1} i_2 \xrightarrow{r_2} e_3 \xrightarrow{-r_3} i_4$. User u_2 clicks song i_2 . Song i_2 is composed of e_3 . e_3 is also a listener of song i_4 . Let's make a song recommendation to u_2 . Consider recommending i_4 to u_2 , u_2 does not click on i_4 , because the second-order neighborhood of u_2 is interested in i_4 .

In a higher-order connectivity graph (such as Fig. 1 b), the relationships between e_{u_1} and its neighbors e_{i_1} , e_{i_3} , e_{i_6} are often ignored. We would like to design an attention mechanism to distinguish the importance of different neighbors by weighting.

3.3 Embedding Layer

The conversion of entities into knowledge graph embedding vectors is an effective way to parameterize entities and relations. The graph structure can be used to represent the higher order propagation between entities. The TransD [42] model considers that the mapping function should be related to entities and relations. It modifies the mapping function by projecting two mapping matrices for head and tail entities. As a result, the confidence score of the triplet is obtained as follows.

$$f_r(h, t) = \|h_{\perp} + r - t_{\perp}\|_{l_1/l_2} \quad (1)$$

Where $h_{\perp} = M_{rh}h$, $t_{\perp} = M_{rt}t$, and $M_{rh} = r_p h_p^T + I^{m \times n}$, $M_{rt} = r_p t_p^T + I^{m \times n}$, $I^{m \times n}$ is the unit matrix, where the head entity and the tail entity are projected with two different mapping matrices.

The mapping matrix of the head entity is determined by the relationship vector r_p together with the head entity mapping vector h_p . While the mapping matrix of the tail entity is determined by the relationship vector r_p together with the tail entity mapping vector t_p .

We often neglect some valid information when entities interact with each other. For example, each entity is contained in multiple triples, such as the tail entity from the

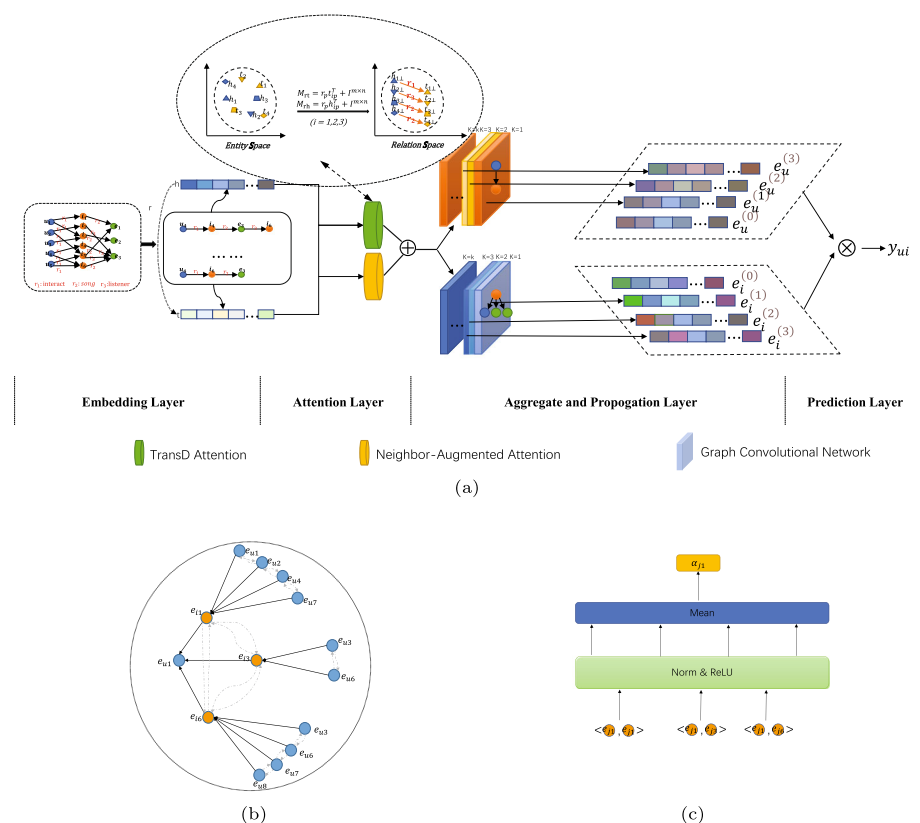


Fig. 1 Neighbor-augmented knowledge graph attention network and sampling strategy. **a** Neighbor-augmented knowledge graph attention network; **b** neighborhood interaction methods between entities; **c** attention model between neighbors

(*Jon*, *Director*, *Iron Man*) triple is Iron Man. While in (*Iron Man*, *Actor*, *Robert*), Iron Man is the head entity, and in (*Jon*, *Actor*, *Friends*), Jon is the head entity. For the head entity h , we use $N_h = \{(h, r, t) \mid (h, r, t) \in G\}$ to denote the set of triples with h as the head node. This information is important for us to explore the first-order connectivity. The relationship between entity h and entity t in the space of relations r can be formulated by the following equation.

$$e_{N_h} = \sum_{(h,r,t) \in N_h} \beta(h, r, t) e_t \quad (2)$$

Where N_h denotes the set of neighbors of entity h .

$$\beta(h, r, t) = (e_{t\perp})^T \tanh(e_{h\perp} + e_r) \quad (3)$$

Where $\beta(h, r, t)$ can be viewed as the importance of entity h to entity t in the relation r -space. $e_{t\perp}$ is the embedding vector of the head entity. $e_{h\perp}$ is the embedding vector of the tail entity. And e_r is the embedding vector of the relation.

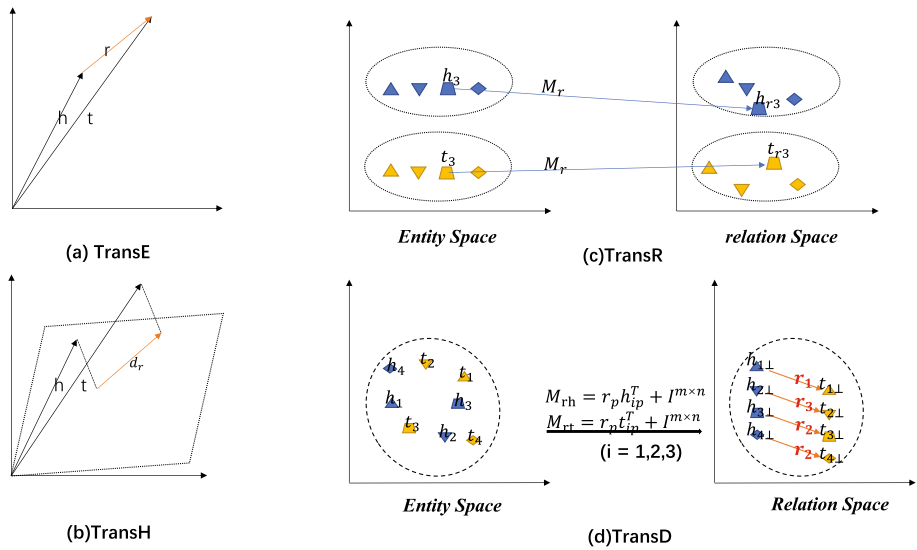


Fig. 2 A graphical illustration of different translation methods in vector space

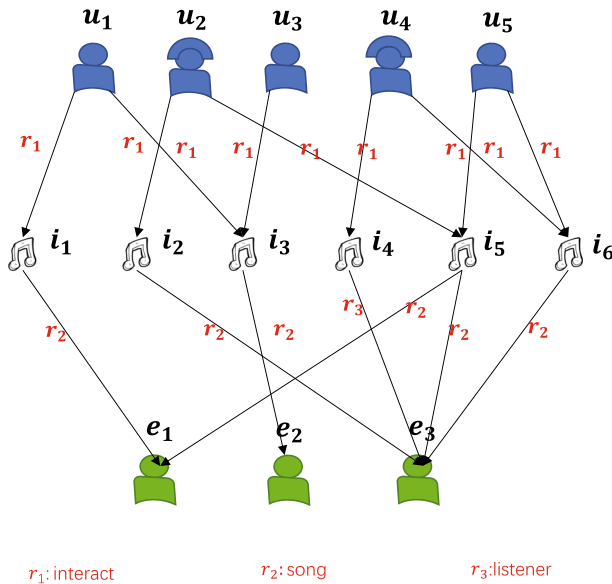


Fig. 3 A higher-order connectivity diagram, the first layer is user u , the second layer is project i (assumed to be a music collection here), and the third layer is other entities. r_1 means that there is an interaction between the user and the project, i.e., the user listens to this music, r_2 means that i is composed and sung by entity e . r_3 means that e_3 is also a listener of i . There exists the case of being both a singer and a listener

We try to use tanh nonlinear activation function because tanh activation function can map the input to $[-1, 1]$. $\beta(h, r, t)$ can be normalized by softmax function.

$$\beta(h, r, t) = \text{softmax}(\beta(h, r, t)) = \frac{\exp(\beta(h, r, t))}{\sum_{(h, r, t) \in N_h} \exp(\beta(h, r, t))} \quad (4)$$

3.4 Neighbor Information Propagation

Similar to NGCF [3], most of the previous work explores the first-order neighbor information, second-order neighbor information, or K-order neighbor information, aggregated by the propagation mechanism of GCN [43, 44]. For example, GC-MC [45] presents the user-item bipartite graph and extracts the user-item interaction information over the spectral domain using GCN. Nevertheless, GC-MC [45] only utilizes the information of first-order neighbors.

NGCF [3] explores higher-order connectivity by extracting user-items as bipartite graphs with their higher-order information, extending the order to the 3rd and 4th order. Although graph neural networks make good use of the information of nodes and edges on the graph, as the order grows, the feature expressions of each node converge, leading to poor results, i.e., over-smoothing. In addition to increasing the number of layers, it is essential to focus on the information between neighbors in the shallow structure.

Similar items play an important role in mining users' preferences, while they are often ignored before. For example, in Fig. 1b, i_1 and i_3 are both first-order neighbors of u_1 , and the relationship between i_1 and i_3 plays an active role in exploring the interested item of u_1 . The first-order neighbors of user u_1 are i_1 , i_2 and i_6 . There is also a potential relationship between all i . We can explore the relationship between their neighbors in the same hop for each entity node.

$$e_{N_h}^{NAA} = \frac{1}{\sqrt{|N_h|}} \sum_{t \in N_h} \alpha_t e_t \quad (5)$$

$$\alpha_t = \text{Softmax} \left(\frac{1}{|N_h|} \sum_{j \in N_h} f(e_t, e_j) \right) \quad (6)$$

$$f(e_t, e_j) = \text{ReLU} \left(\frac{e_t^T e_j}{\|e_t\| \cdot \|e_j\|} \right) \quad (7)$$

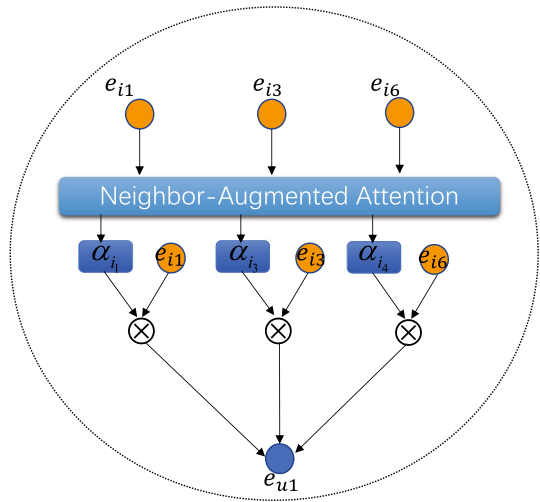
Where $j \in N_h$. We can use the embedding information of neighbor j , similar to neighbor t . α_t represents the importance of relationship r to entity h . We use $\frac{1}{\sqrt{|N_h|}}$ to measure the importance of different entities t . Figure 4 is a visual representation of Eq. (8).

We define the higher-order connectivity of the model as follows:

$$e_{N_h}^{(K)} = f(e_{N_h}^{(K)}, e_{N_h}^{NAA(K)}) \quad (8)$$

Where $e_{N_h}^{(K)}$ is the neighbor information of h aggregated by GCN. It is composed of two parts. $e_{N_h}^{(K)} = \sum_{(h, r, t) \in N_h} \beta(h, r, t) e_t^{(K-1)}$, e_t^{K-1} is the embedding expression obtained by aggregating entity t through $K-1$ layers. $e_h^{(0)}$ is the information obtained by random initialization. $f(\cdot)$ is the sum operation here.

Fig. 4 The operation mechanism of neighbor-augmented attention in the triplet (h, r, t)



3.5 Aggregation and Propagation Layer

3.5.1 Aggregation Layer

Since graph convolutional neural networks can extract rich and effective information from graphically structured data, they are often used in recommendation learning tasks. However, feature decomposition of large matrices is required in graph convolutional neural networks. To avoid the operation of feature decomposition, GCNs constrain the convolution kernel and simplify the graph convolution to a graph propagation. It is validated that aggregating the obtained embedding vectors using GCN [37] is effective and much better than the traditional model. The commonly used aggregation methods of GCN are mean aggregation, maximum pooling aggregation, LSTM aggregation, etc. GraphSAGE [46] provides a more concise expression of aggregation, i.e., the features of each node are aggregated to the central node utilizing nonlinear transformation and nonlinear activation. But SGC [47] demonstrates that the additional complexity of GCNs can be reduced by iteratively eliminating the nonlinearity between GCN layers.

GCN Aggregator

$$f_{GCN} = \sigma(W(e_h + e_{\mathfrak{N}_h})) \quad (9)$$

GraphSAGE Aggregator

$$f_{GraphSAGE} = \sigma(W(e_h \| e_{\mathfrak{N}_h})) \quad (10)$$

Bi-Interaction Aggregator

$$f_{Bi-Interact} = \sigma(W_1(e_h + e_{\mathfrak{N}_h}) + W_2(e_h \odot e_{\mathfrak{N}_h})) \quad (11)$$

LightGCN [10] removes the linear transformation W as well as the nonlinear activation. Inspired by LightGCN [10], we also remove W and σ in this paper.

LGC Aggregator

$$f_{LGC} = Identity(e_h + e_{\mathfrak{N}_h}) \quad (12)$$

The Identity transformation is adopted as the aggregation function in the developed model.

Bi-LGC Aggregator

$$f_{Bi-LGC} = Identity((e_h + e_{\mathbb{N}_h}) + (e_h \odot e_{\mathbb{N}_h})) \quad (13)$$

We use mean function to average the input information and Bi aggregation to enrich the expression.

3.5.2 Model Prediction

With aggregating the information of the K -hop neighbor nodes, we obtain multiple expressions for the central node u , $e_u^{(1)}, e_u^{(2)}, e_u^{(3)}, \dots, e_u^{(K)}$. Similarly, we can also obtain the expression for item i , $e_i^{(1)}, e_i^{(2)}, e_i^{(3)}, \dots, e_i^{(K)}$. The inner products of users and items are calculated as the predicted value.

$$\hat{y}(u, i) = \langle e_u^*, e_i^* \rangle \quad (14)$$

Where $e_u^* = e_u^{(0)} \| e_u^{(1)} \| e_u^{(2)} \| \dots \| e_u^{(K)}$, $e_i^* = e_i^{(0)} \| e_i^{(1)} \| e_i^{(2)} \| \dots \| e_i^{(K)}$.

4 Optimization

4.1 Loss Function

Let \mathcal{L}_{KG} represent the loss function of KG part.

$$\mathcal{L}_{KG} = \sum_{(h,r,t,t') \in \tau} -\ln \sigma(g(h, r, t) - g(h, r, t')) \quad (15)$$

Where $\tau = \{(h, r, t, t') \mid (h, r, t) \in \mathcal{G}, (h, r, t') \notin \mathcal{G}\}$, h is the head entity vector. r is the relation vector. t is the tail entity vector. (h, r, t') is a corrupted triple with a random replacement of the tail entity vector, which is used to train the model. It is also possible to replace the head entity vector h in a random way. Here we replace the tail entity vector t .

$$\mathcal{L}_{CF} = \sum_{(u,i,j) \in O} -\ln \sigma(\hat{y}(u, i) - \hat{y}(u, j)) \quad (16)$$

Where $O = \{(u, i, j) \mid (u, i) \in \mathcal{R}^+, (u, j) \in \mathcal{R}^-\}$ refers to the training set, \mathcal{R}^+ is the observable (positive) set between u and i , and \mathcal{R}^- is the set of unobserved interactions randomly sampled by negative sampling. $\sigma(\cdot)$ is the sigmoid activation function.

$$\mathcal{L} = \mathcal{L}_{KG} + \mathcal{L}_{CF} + \lambda \|\theta\|_2^2 \quad (17)$$

We use a multi-task training approach to divide the overall loss into KG and CF, where $\Theta = \{E, \forall l \in \mathcal{R}, W_l^{(l)}, \forall l \in \{1, \dots, L\}\}$ is the parameter set part of the model. E is the vector expression of all entities. l_2 regularization is used to reduce over-fitting. λ is a weight.

4.2 Training

The mini-batch Adam optimizer is adopted to optimize the loss function. The Adam optimizer can calculate an adaptive learning rate for each parameter. If the data is sparse, the gradient

would be sparse. In this case, Adam is a good choice because it can converge to the global optimal point fast. In the training process, we randomly sample a batch of (h, r, t, t') , and update all the node embedding expressions. In CF part, we randomly sample a batch of (u, i, j) , and after propagating K layers. We can get their embedding expressions, and finally update the parameters of the model using the gradient of the prediction loss.

5 Experiments

5.1 Datasets

We used the following datasets to validate the proposed model: Last-FM, Yelp2018, MovieLens, and Book-Crossing.

Last-FM is a popular music dataset that contains user comments on music items. We select the dataset with a timestamp from June 2015 to July 2015 and use the same 10-core setting [38]. It can be obtained from the website: <https://grouplens.org/datasets/hetrec-2011/>.

Yelp2018 is selected from the 2018 version of the Yelp challenge, and the data contains commercial buildings such as restaurants and bars, which we treat them as commodities. Similarly, a 10-core setting is used [38]. It can be obtained from the website: <https://www.yelp.com/dataset>.

MovieLens-20M is a typical dataset in movie recommendations and contains about 20 million rating data. It can be obtained from the website: <https://grouplens.org/datasets/movielens/20m/>.

Book-Crossing contains about 1 million ratings from 0 to 10 in the book community. It can be obtained from website: [http://www2.informatik.uni-freiburg.de/~sim\\$chiegler/BX/](http://www2.informatik.uni-freiburg.de/~sim$chiegler/BX/).

In addition to extracting user-item interactions, it is necessary to construct knowledge graphs for each dataset. Items are mapped to entities by title matching in KGAT [38]. To ensure the quality of KG, we filter out the uncommon entities and keep the entities that appeared at least 50 times. For each dataset, we randomly select 80% as the training set, 10% as the test set, and the remaining 10% as the validation set to adjust the hyperparameters. We select observable instances of user-item interactions as positive instances. We use negative sampling mining to pair negative items which users have never associated with. The details of the two datasets are shown in Table 1.

Table 1 Details of the datasets

	Last-FM	Yelp2018
#User	23566	45919
#Item	48123	45538
#Interactions	3034796	1185068
#Entities	58266	90961
#Relations	9	42
#Triplets	464567	1853704

5.1.1 Evaluation Metrics

In order to compare with the other methods, we calculate Recall and NDCG on Last-FM and Yelp2018 datasets. It is the same as KGAT [38]. Similarly, AUC and F_1 are calculated on MovieLens-20M, Book-Crossing, and Last-FM, which is the same as KGCN [37].

We adopt Recall and NDCG as evaluation metrics for our first controlled test set. To evaluate click-through rate prediction, we use AUC and F_1 as evaluation metrics.

5.2 Experimental Details

The model is achieved by TensorFlow. The embedding size of the model was set to 64, which is the same as KGAT [38]. We select Adam as optimizer. The batch size is 1024. And the Xavier initial values are used to initialize the model parameters. A grid search is used to select hyperparameters. The learning rate is adjusted in $\{0.05, 0.01, 0.005, 0.001\}$. The L_2 normalization is searched in $\{10^{-5}, 10^{-4}, \dots, 10, 100\}$, and the dropout values are adjusted in $\{0.0, 0.1, \dots, 0.9\}$. In addition, we also set an early stopping strategy. The program would stop if the Recall indicator did not increase for 30 consecutive epochs. To explore the effect of third-order connectivity on the effect of the model, we set the depth $K = 3$ and the dimensions of hidden layers to 64, 32 and 16, respectively.

5.2.1 Baselines

The proposed method is compared to well-known and SOTA models. PER [48] treats the problem of knowledge mapping as a heterogeneous information network, and PER extracts meta-path-based features to represent the connections between users and items. CKE [29] integrates CF with structural, visual, and text-based knowledge. RippleNet [23] combines regularization and path-based approaches to propagate users' preferences over KG. KGAT [38] is a model that uses TransE [15] as the KG constraint part, fuses CF, and uses GAT for aggregation. KGCN [37] uses GCN for aggregation on KG graphs.

CKE [29] is a typical rule-based approach that utilizes TransR-derived semantic embeddings with enhanced matrix decomposition. RippleNet [23] combines regularization and path-based approaches that enrich the user representation by adding entries on each user's root path, which propagates the users' preferences on KG for a recommendation. PER treats KG as a heterogeneous information network, extracting meta-path-based features to represent the connections between users and items. KGAT [38] considers connecting two items with one or more link attributes in the hybrid structure of KG and user-item graph as KGCN [37] effectively captures the correlation between items by mining their related attributes on the KG.

After constructing the graph structure, information between neighbors is also crucial. Our method models higher-order connections end-to-end. It recursively propagates neighbors embeddings to refine node embeddings. Attention mechanism assigns different importances to neighbors.

Our models outperform the most advanced methods, such as KGCN [37], a model that uses entity triples in an aggregated way using graph convolutional neural networks. KGCN [37] uses the inner product between the head entity h and the relation r as the attention mechanism, while our NKGAT uses the TransD [42] model and the hidden relations between the neighbors of entities as the attention mechanism.

Table 2 Top K of different methods

	Last-FM		Yelp2018	
	Recall@20	NDCG@20	Recall@20	NDCG@20
CKE	0.0736	0.1184	0.0657	0.0805
RippleNet	0.0791	0.1238	0.0664	0.0822
KGCN	0.0855	0.1311	0.0695	0.0823
KGAT	0.0870	<u>0.1325</u>	0.0712	<u>0.0867</u>
KGNN-LS	<u>0.0880</u>	0.1301	<u>0.0722</u>	0.0851
CKAN	0.0864	0.1321	0.0704	0.0852
NKGAT	0.0920	0.1406	0.0763	0.0917

The best indicators in references are marked with underlines. The best indicators are marked in bold font

Table 3 The results of AUC and $F1$ in CTR prediction

	MovieLens-20M		Book-crossing		Last-FM	
	AUC	F1	AUC	F1	AUC	F1
CKE	0.924	0.871	0.677	0.611	0.744	0.673
RippleNet	0.968	0.912	0.715	0.650	0.780	0.702
KGCN	<u>0.978</u>	<u>0.932</u>	0.738	0.631	0.794	0.719
KGAT	0.976	0.928	0.731	0.654	0.829	0.742
KGNN-LS	0.975	0.929	0.676	0.631	0.805	0.722
CKAN	0.976	0.929	<u>0.753</u>	<u>0.673</u>	<u>0.842</u>	<u>0.769</u>
NKGAT -sum	0.980	0.939	0.76	0.707	0.85	0.773
NKGAT -concat	0.981	0.939	0.763	0.706	0.849	0.769
NKGAT -BiLGC	0.981	0.939	0.762	0.707	0.85	0.771

The best indicators in references are marked with underlines. The best indicators are marked in bold font

5.3 Results

In order to compare with KGAT [38], Recall and NDCG are selected to evaluate different methods on two datasets: Last-FM and Yelp2018. The results are shown in Table 2. The proposed NKGAT achieves the best performance.

In order to compare with KGCN [37], we consider the CTR problem. AUC and F1 are selected on MovieLens-20M, Book-Crossing, and Last-FM datasets. The results are shown in Table 3. The results of all the baselines are obtained from KGAT [38], KGCN [37] and CKAN [39].

As shown in Table 3, NKGAT outperforms the state-of-the-art baseline in all metrics. Table 3 demonstrates the effectiveness and advancement of NKGAT.

According to our analysis, three reasons probably led to this progress: (1) The importance of relationships to entities is considered. (2) Similar neighbors would be important for exploring users' preferences, we mine the collaboration information between neighbors in a neighborhood. (3) In the knowledge graph step, we use TransD [42] instead of other models in the Trans family. It can reduce the model's parameters, which greatly optimizes its performance. The results indicate that using the information between neighbors of the same hop is helpful.

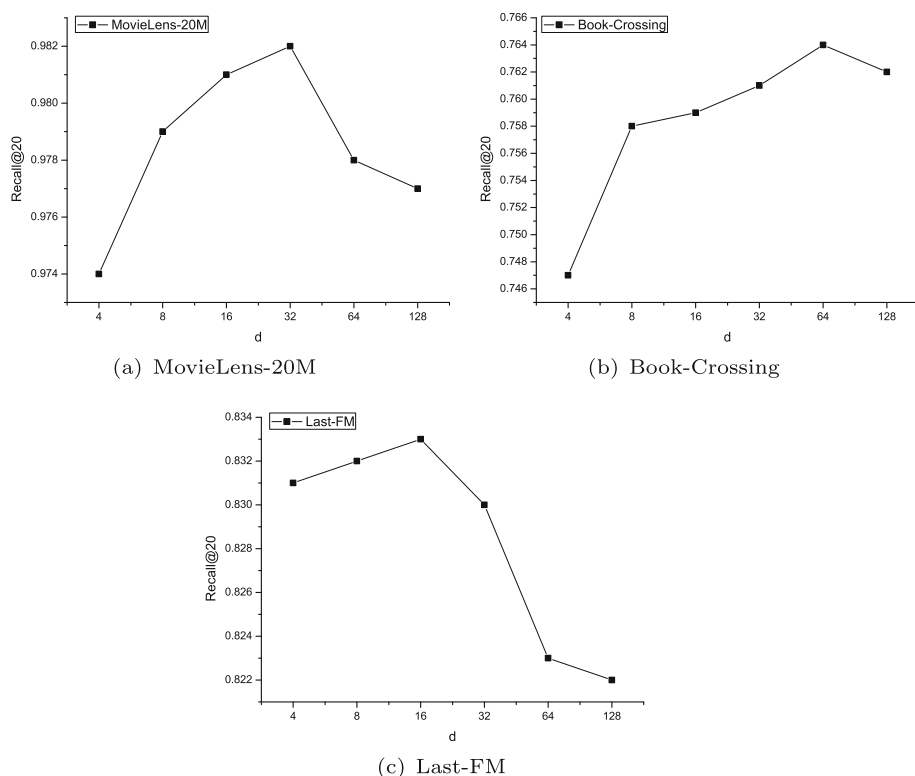


Fig. 5 AUC of different dimensions on all three datasets

In addition, NKGAT verifies the effectiveness of attention. In general, CKE [29] only uses their embedding of associated entities. NKGAT performs better than Ripple, because Ripple uses relational information to explore user preferences without considering the weight of the user's neighbors. Comparing CKAN [39] with KGNN-LS [37], it is found that CKAN [39] is usually better than KGNN-LS [37], only the Recall@20 on Last-FM is not competitive. The reason may be that in addition to model project knowledge, CKAN [39] also further considers user-item collaborative signals for recommendations. KGCN [37] and KGAT [38] combine KG with graph neural networks, but do not consider the relevance between different neighbors of the central entity.

There are three variants of NKGAT summarized in Table 3, namely *SUM*, *CONCAT*, and *BiLGC*, where *BiLGC* is the feature interaction information of e_h and e_{Nh} introduced in the aggregation stage, and the operation of nonlinear activation and linear transformation is eliminated. This makes the propagated information sensitive to the similarity between neighbors and central entities.

We also do some experiment with different hidden layer dimensions. The experimental results are shown in Fig. 5. For MovieLens-20M, the AUC is optimal at 32. For Book-Crossing, the AUC is optimal at 64. For Last-FM, the AUC is optimal at 16. In general, a larger hidden layer dimension can encode more information, but too much over-coding may be overfitting due to the influence of the dataset. Figure 5 demonstrates the effect of different dimensions on AUC.

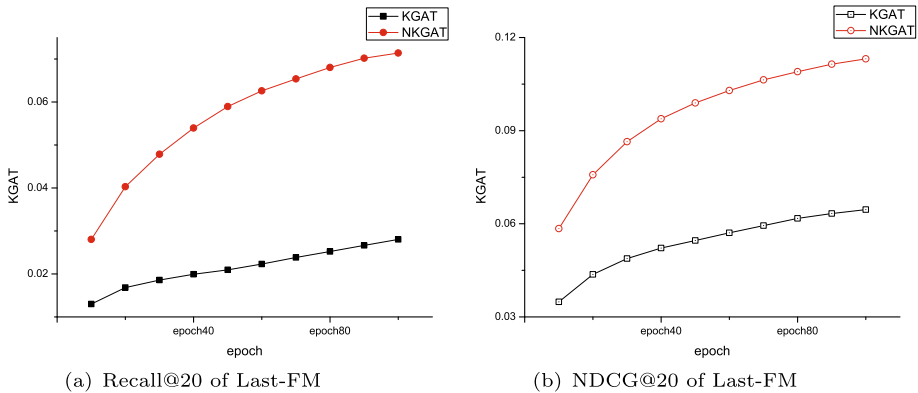


Fig. 6 The result of *Recall@20* and *NDCG@20* in top-*K* recommendation of Last-FM

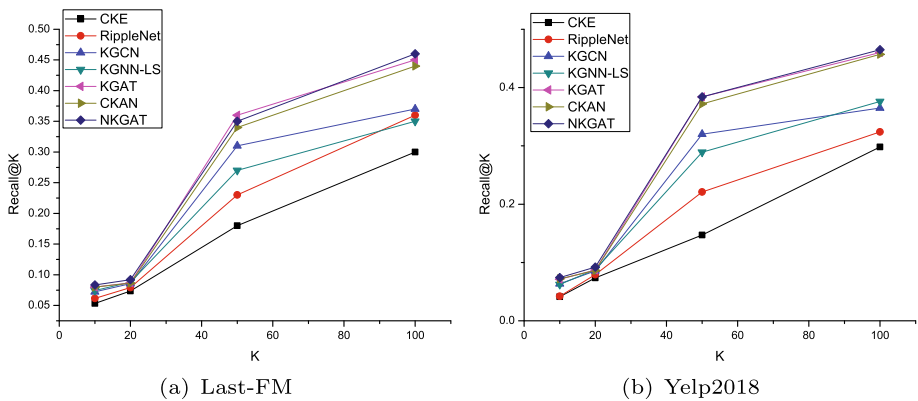


Fig. 7 The result of *Recall@K* in top-*K* recommendation

Figure 6 shows that in the first 100 batches in the iteration, our model can converge about three times faster than KGAT [38]. It means that the convergence speed is greatly improved. Figure 7 shows the effect of the model with different *K*.

5.4 Ablation Study

To examine the effectiveness of the TransD [42] module and the neighborhood attention mechanism, we conducted ablation experiments on both modules, using Recall@20 as the metrics. The experimental results are shown in Table 4.

In the ablation experiments, we replace the TransD [42] module in NKGAT with TransR [17] and TransE [15], respectively. The results show that using TransD [42] is better than TransE [15] and TransR [17]. This indicates the importance of mapping different entities to different spaces. There is also a decrease in the effect after removing our Neighborhood-Augmented Attention layer (NAA) based on NKGAT. It also shows the effectiveness of our model and the importance of the relationship between neighbors within the same hop. The experiments show that the model of TransD+NAA achieves a Recall of 0.0920 and NDCG of 0.1406 on Last-FM. The Recall achieves 0.0763 and NDCG is 0.0917 on Yelp2018.

Table 4 Ablation experiment

	Last-FM		Yelp2018	
	Recall@20	NDCG@20	Recall@20	NDCG@20
TransR+NAA	0.0899	0.1389	0.0742	0.0892
TransE+NAA	0.0854	0.1374	0.0735	0.0874
TransD	0.0901	0.1396	0.0750	0.0901
TransD+NAA	0.0920	0.1406	0.0763	0.0917

The best indicators in references are marked with underlines. The best indicators are marked in bold font

6 Conclusion

In this work, we utilize the relationships and attributes in knowledge graph employ entity attributes and higher-order relationships between entities to achieve more accurate recommendations. In addition, we introduce Neighborhood Augmented Attention, which can distinguish the different importance of neighbors in the same neighborhood for the same entity. We use a graph convolutional neural network to propagate embeddings from node neighbors and update the current node representation. The effectiveness of proposed model is validated on four datasets.

In the future, we would like to apply our model in more general graph data structures and test it on large and diverse datasets. In addition, we would like to consider the following questions for knowledge graph-based recommendation systems: How to make the knowledge graph explanatory in more fields? How to quickly and accurately update the knowledge graph to reflect the changes in the real world?

Acknowledgements This work is supported by National Natural Science Foundations of China, No. 61703088 and 62173101, “the Fundamental Research Funds for the Central Universities”, N2105009 and N180503017.

Data Availability All data generated or analysed during this study are included in this published article. The datasets generated during and/or analyzed during the current study are available in the repository, <https://github.com/cuihao798448708/>.

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.

References

1. He X, He Z, Song J, Liu Z, Jiang Y-G, Chua T-S (2018) Nais: neural attentive item similarity model for recommendation. *IEEE Trans Knowl Data Eng* 30(12):2354–2366
2. 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
3. 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
4. Zhang L, Li Z, Sun X (2021) Iterative rating prediction for neighborhood-based collaborative filtering. *Appl Intell* 1–13
5. Koren Y, Bell R, Volinsky C (2009) Matrix factorization techniques for recommender systems. *Computer* 42(8):30–37

6. Sun X, Zhang H, Wang M, Yu M, Yin M, Zhang B (2020) Deep plot-aware generalized matrix factorization for collaborative filtering. *Neural Process Lett* 52(3):1983–1995
7. Rendle S (2010) Factorization machines. In: 2010 IEEE international conference on data mining, pp 995–1000. IEEE
8. 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
9. Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G (2008) The graph neural network model. *IEEE Trans Neural Netw* 20(1):61–80
10. 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
11. Sun J, Zhang Y, Guo W, Guo H, Tang R, He X, Ma C, Coates M (2020) Neighbor interaction aware graph convolution networks for recommendation. In: Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval, pp 1289–1298
12. Cao Y, Hou L, Li J, Liu Z (2018) Neural collective entity linking. *arXiv preprint [arXiv:1811.08603](https://arxiv.org/abs/1811.08603)*
13. Cao Y, Hou L, Li J, Liu Z, Li C, Chen X, Dong T (2018) Joint representation learning of cross-lingual words and entities via attentive distant supervision. *arXiv preprint [arXiv:1811.10776](https://arxiv.org/abs/1811.10776)*
14. Zhao H, Yao Q, Li J, Song Y, Lee DL (2017) Meta-graph based recommendation fusion over heterogeneous information networks. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, pp 635–644
15. Bordes A, Usunier N, Garcia-Duran A, Weston J, Yakhnenko O (2013) Translating embeddings for modeling multi-relational data. In: *Neural information processing systems (NIPS)*, pp 1–9
16. Wang Z, Zhang J, Feng J, Chen Z (2014) Knowledge graph embedding by translating on hyperplanes. In: *Proceedings of the AAAI conference on artificial intelligence*, vol. 28
17. Lin Y, Liu Z, Sun M, Liu Y, Zhu X (2015) Learning entity and relation embeddings for knowledge graph completion. In: *Proceedings of the AAAI conference on artificial intelligence*, vol. 29
18. Catherine R, Cohen W (2016) Personalized recommendations using knowledge graphs: a probabilistic logic programming approach. In: *Proceedings of the 10th ACM conference on recommender systems*, pp 325–332
19. Hu B, Shi C, Zhao WX, Yu PS (2018) Leveraging meta-path based context for top-n recommendation with a neural co-attention model. In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp 1531–1540
20. Ma W, Zhang M, Cao Y, Jin W, Wang C, Liu Y, Ma S, Ren X (2019) Jointly learning explainable rules for recommendation with knowledge graph. In: *The world wide web conference*, pp 1210–1221
21. Sun Z, Yang J, Zhang J, Bozzon A, Huang L-K, Xu C (2018) Recurrent knowledge graph embedding for effective recommendation. In: *Proceedings of the 12th ACM conference on recommender systems*, pp 297–305
22. 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
23. Wang H, Zhang F, Wang J, Zhao M, Li W, Xie X, Guo M (2018) Ripplet: 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
24. Ai Q, Azizi V, Chen X, Zhang Y (2018) Learning heterogeneous knowledge base embeddings for explainable recommendation. *Algorithms* 11(9):137
25. Cao Y, Wang X, He X, Hu Z, Chua T-S (2019) Unifying knowledge graph learning and recommendation: towards a better understanding of user preferences. In: *The world wide web conference*, pp 151–161
26. Huang J, Zhao WX, Dou H, Wen J-R, Chang EY (2018) Improving sequential recommendation with knowledge-enhanced memory networks. In: *The 41st international ACM SIGIR conference on research & development in information retrieval*, pp 505–514
27. Wang C, Zhang M, Ma W, Liu Y, Ma S (2020) Make it a chorus: knowledge-and time-aware item modeling for sequential recommendation. In: *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pp 109–118
28. Wang H, Zhang F, Xie X, Guo M (2018) Dkn: deep knowledge-aware network for news recommendation. In: *Proceedings of the 2018 world wide web conference*, pp 1835–1844
29. Zhang F, Yuan NJ, 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
30. Wang P, Fan Y, Xia L, Zhao WX, Niu S, Huang J (2020) Kerl: a knowledge-guided reinforcement learning model for sequential recommendation. In: *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pp 209–218

31. Wang X, Xu Y, He X, Cao Y, Wang M, Chua T-S (2020) Reinforced negative sampling over knowledge graph for recommendation. In: Proceedings of the web conference 2020, pp 99–109
32. Xian Y, Fu Z, Muthukrishnan S, De Melo G, Zhang Y (2019) Reinforcement knowledge graph reasoning for explainable recommendation. In: Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval, pp 285–294
33. Zhao K, Wang X, Zhang Y, Zhao L, Liu Z, Xing C, Xie X (2020) Leveraging demonstrations for reinforcement recommendation reasoning over knowledge graphs. In: Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval, pp 239–248
34. Zhou S, Dai X, Chen H, Zhang W, Ren K, Tang R, He X, Yu Y (2020) Interactive recommender system via knowledge graph-enhanced reinforcement learning. In: Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval, pp 179–188
35. Zhang J, Ma C, Zhong C, Mu X, Wang L (2021) Mbpi: mixed behaviors and preference interaction for session-based recommendation. *Appl Intell* 1–13
36. Jin J, Qin J, Fang Y, Du K, Zhang W, Yu Y, Zhang Z, Smola AJ (2020) An efficient neighborhood-based interaction model for recommendation on heterogeneous graph. In: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining, pp 75–84
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 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
39. Wang Z, Lin G, Tan H, Chen Q, Liu X (2020) Ckan: Collaborative knowledge-aware attentive network for recommender systems. In: Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval, pp 219–228
40. Ma M, Na S, Wang H, Chen C, Xu J (2021) The graph-based behavior-aware recommendation for interactive news. *Appl Intell* 1–17
41. Pujahari A, Sisodia DS (2021) Preference relation based collaborative filtering with graph aggregation for group recommender system. *Appl Intell* 51(1):1–15
42. Ji G, He S, Xu L, Liu K, Zhao J (2015) Knowledge graph embedding via dynamic mapping matrix. In: Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (Volume 1: Long Papers), pp 687–696
43. Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. *arXiv preprint [arXiv:1609.02907](https://arxiv.org/abs/1609.02907)*
44. Hamilton WL, Ying R, Leskovec J (2017) Inductive representation learning on large graphs. *arXiv preprint [arXiv:1706.02216](https://arxiv.org/abs/1706.02216)*
45. Berg Rvd, Kipf TN, Welling M (2017) Graph convolutional matrix completion. *arXiv preprint [arXiv:1706.02263](https://arxiv.org/abs/1706.02263)*
46. Veličković P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y (2017) Graph attention networks. *arXiv preprint [arXiv:1710.10903](https://arxiv.org/abs/1710.10903)*
47. Wu F, Souza A, Zhang T, Fifty C, Yu T, Weinberger K (2019) Simplifying graph convolutional networks. In: International conference on machine learning, pp 6861–6871. PMLR
48. Yu X, Ren X, Sun Y, Gu Q, Sturt B, Khandelwal U, Norick B, Han J (2014) Personalized entity recommendation: a heterogeneous information network approach. In: Proceedings of the 7th ACM international conference on web search and data mining, pp 283–292

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.