GATKR: Knowledge Graph Recommendation Algorithm Incorporating Graph Attention Networks

Huangyu Sheng,Xuefeng Fu*,Yikun Xiong, Hao Hu, and Taotao Wang

Abstract—**Knowledge graphs (KGs) serve as supplementary data sources for recommendation models, effectively addressing challenges such as cold-start scenarios and data sparsity. Existing KG-based recommender systems ignore the importance of different entities when processing KG, resulting in biased modeling of user and item characteristics; meanwhile, useless information in the Knowledge graph also affects the performance of recommendations. To address such problems, we propose an algorithm named GATKR (graph attention network in knowledge graph recommender), which uses graph attention networks (GATs) to traverse the interaction graph and knowledge graph and apply filters to remove irrelevant information from the knowledge graph. This algorithm first uses a semantic matching model for embedding, then aggregates neighborhood features using GAT, complements user and item features using the attention function as a weight for neighborhood feature selection, and finally splices and inner products the features obtained from embedding and those from GAT to get user and item interaction probabilities. With a maximum improvement of 4% in the cumulative gain of normalized discount, the algorithm's efficacy is confirmed on four datasets (Amazon-Book, Yelp2018, Last-FM, Alibaba).**

Index Terms—**Graph Attention Network; Knowledge Graph; Recommender System; Embedding Filtering**

I. Introduction

THE development of the information era has led to a massive emergence of information, resulting in severe information overload. Search engines or recommender systems are often used to cope with information overload. Search engines provide information that meets the user's requirements by leveraging keywords, particularly in cases where the user has a specific query. In contrast, recommender systems do not require keywords to recommend information that is of interest to the user. This feature significantly enhances the user experience and has become a prevalent feature in various domains, including e-commerce platforms, music, and short video platforms.

There are three categories for traditional recommendations: content-based, collaborative filter-based and hybrid [1,2]. Content-based recommendation extracts the features of items from the item descriptions, matches them with the features of items that users have interacted with, and recommends the items with the same features to users. Collaborative filter-based recommendation begins with the user and identifies a group of users with similar interests to suggest items they have interacted with or items similar to those the user has interacted with. The hybrid recommendation combines the two methods, taking advantage of their strengths and overcoming weaknesses. Traditional recommendation requires a large number of interaction records, so no recommendation can be made if there are missing user or item interaction records.

KG-based recommendation algorithms address some of the problems in traditional recommendation. KG [3,4] is a large-scale graph data of cross-domain information [5], which is structured to store information via a triplet of (head entity, relation, tail entity). The head entity and tail entity are nodes in the graph, and the relation is a directed edge from the head entity to the tail entity. Recommendation algorithms based on KG can be divided into the following three types according to the different processing of the triplets.

The first is an embedding-based recommendation algorithm, knowledge graph embedding using vectors to represent entities and relations in the triplets, enriching user and vector representations through vectors. The second is path-based, with the head entity and tail entity getting path similarity through meta-path and then getting the user's preference for recommendation with the help of path similarity. The third is graph neural network-based, which integrates the advantages of embedding algorithms and uses graph neural network (GNN) to pass messages, improving the interpretability of the recommendation.

Inspired by the knowledge graph attention network (KGAT) [6] model, we propose a KG-based recommendation algorithm, graph attention network in knowledge graph recommender (GATKR), which incorporates GAT, with the assistance of ConvKB [7] for knowledge graph embedding. GAT for message propagation and aggregation of neighbor information while filtering out useless information using filters to improve the performance and interpretability of recommendations.

The main contributions of this paper are as follows:

(1) We propose a recommendation algorithm that effectively utilizes knowledge graph information through GAT. It effectively models node information while filtering know-

Manuscript received July 18, 2024; revised February 3, 2025. Research on this work was partially supported by the grants from the National Science Foundation of China (No. 61762063).

Xuefeng Fu is an Associate Professor at Nanchang Institute of Technology (NIT), Nanchang 330099, China. (Corresponding author, tel: 86-18170936669, email: fxf@nit.edu.cn)

Huangyu Sheng is a graduate student of Nanchang Institute of Technology (NIT), Nanchang, China. (email: 1271425661@qq.com)

Yikun Xiong is a graduate student of Nanchang Institute of Technology (NIT), Nanchang, China. (email: 1326168741@qq.com)

Hao Hu is a graduate student of Nanchang Institute of Technology (NIT), Nanchang, China. (email: 1847800736@qq.com)

Taotao Wang is a graduate student of Nanchang Institute of Technology (NIT), Nanchang, China. (email: 1440338590@qq.com)

ledge graph noise.

(2) In our studies, we tested GATKR on four different datasets and showed that it performs better than the current state-of-the-art baselines.

II. Related Work

KG-based recommendation algorithms are divided into three classes: embedding-based, path-based and GNN-based.

Embedding-based recommendation algorithms process the KG with knowledge graph embedding, which converts entities and relationships into a low-dimensional vector representation [8]. Zhang et al.'s collaborative knowledge base embedding (CKE) [9] model, which makes joint training of interaction graph and KG, extracts KG by TransR [10] to obtain the structured knowledge. Additionally, a self-encoder is utilized to extract textual knowledge, and a convolutional network is employed to extract visual knowledge. These components collectively constitute the item vectors. Subsequently, these item vectors are integrated with user vectors to yield prediction outcomes through a collaborative filtering approach. A notable drawback of the CKE model is its neglect of the connectivity between different triplets, treating all the triplets in the KG as a single individual, which lacks wholeness. The multi-task learning for knowledge graph enhanced recommendation (MKR) [11] model proposed by Wang et al. alternates between training the KG and the recommender system. The recommender part of the model learns the click rate, while the KG part learns to predict the tail node. The model utilizes cross-feature-sharing units to facilitate the exchange of information between the two parts. This approach addresses the sparsity problem to a certain extent, but the lack of connectivity persists.

Path-based recommendation algorithms get the user's interest preferences through meta-paths and path similarity. Wang et al. proposed a knowledge path recurrent network (KPRN) [12] that encodes the elements on the paths with LSTM, captures the semantic relationships between the entities, and feeds them into a multilayer perceptron (MLP) to get the final score. The selection of meta-paths constitutes a critical aspect of path-based recommendation algorithms, necessitating expertise in relevant domains for their design. A notable aspect of the proposed framework is the non-universality of meta-paths across different domains, a requirement that poses a substantial challenge to designers.

GNN-based recommendation algorithms [13] are differentiated from the other two by focusing more on messaging in the KG and using GNN to process the KG. Knowledge graph convolutional networks (KGCN) [14], proposed by Wang et al., combine a function of user vectors and relation vectors as the messaging weights, which puts more focus on the user's attention than using relationship vectors directly. The weights of the message pass and the inner product of the neighbor nodes are used to obtain the final item vector, and the item vector and the user vector are the inner products to get the predicted click rate via the activation function. Sha et al.'s attentive knowledge graph embedding (AKGE) [15] involves the extraction of subgraphs from user-item pairs. These subgraphs are then pre-trained using knowledge graph embedding, followed by the aggregation of neighbor nodes through an attention mechanism[16]. The representation of items is updated using a gating mechanism, and prediction is made via an artificial neural network (MLP).

In summary, the GNN-based recommendation algorithm demonstrates the most significant efficacy in extracting features from the KG and graph structure information from the KG. Embedding-based recommendation algorithms overlook the graph structure information, whereas path-based recommendation algorithms face more significant challenges in designing meta-paths. Our proposed model is intended to be optimized and extended for recommendation algorithms in GNN.

III. Problem Formulation

Following the dominant recommendation models, we use a combination of user-item interaction graphs and knowledge graphs and describe both the inputs and the outputs of the experiments.

User-Item Interaction Graph: The interaction between the user and the item is divided into display interaction and implicit interaction. The display interaction can get the user's score of the item, and according to the score, we can know whether the user likes the item or not, for example, film rating, product rating, and so on. Implicit interaction can only mean an interaction exists between the user and the item, for example, purchase, click, and so on. In this paper, the interaction in the user-item interaction graph G_1 is implicit interaction, the interaction records in G_1 are $\{(u, y_{ui}, i)|u \in U, i \in I\}$, U is the set of users, and I is the set of items, when $y_{ui} = 1$ means that interaction exists between user u and item i, when $y_{ui} = 0$ means that interaction does not exist between user u, and item i.

Knowledge Graph: It stores additional information about the items. In this paper, each record of the KG G_2 is a triplet, $G_2 = \{(h, r, t)|h, t \in E, r \in R\}$, E is the set of entities and R is the set of relations. The set of items I in G_1 is a subset of the set of entities E. To correspond entities to items, we create $D = \{(\mathrm{i}, \mathrm{e})|i \in I, e \in E\}$, which represents the entity corresponding to an item.

The data in both user-item interaction graph G_1 and KG G_2 can be represented in the form of triplets. We combine the two graphs into a collaborative knowledge graph G, $G = \{(h, r, t)|\ h, t \in E',\ r \in R'\}$, $E' = E \cup U$, $R' = R \cup \{interact\}$, and $\{interact\}$ is the interaction in G_1.

The task of this paper is to predict the probability of interaction between a user and a not-interacted item using a collaborative knowledge graph.

Input: A knowledge graph G_2 and a user-item interaction graph G_1 combine into a collaborative knowledge graph G.

Output: The probability that user u interacts with item i.

IV. The Proposed GATKR Framework

We propose the incorporation of a filter-based attention mechanism into the knowledge graph recommendation algorithm. The model architecture, shown in Figure 1, consists of three main parts: (1) Embedding representation layer, which uses parameterization to produce vector representations of triplet entities and relations. (2) Attention propagation layer, which propagates the entity representation recursively. Each propagation updates the entities using neighbors and filters to filter information and reduce the influence of noise. (3) Prediction layer, user and item vectors are aggregated to

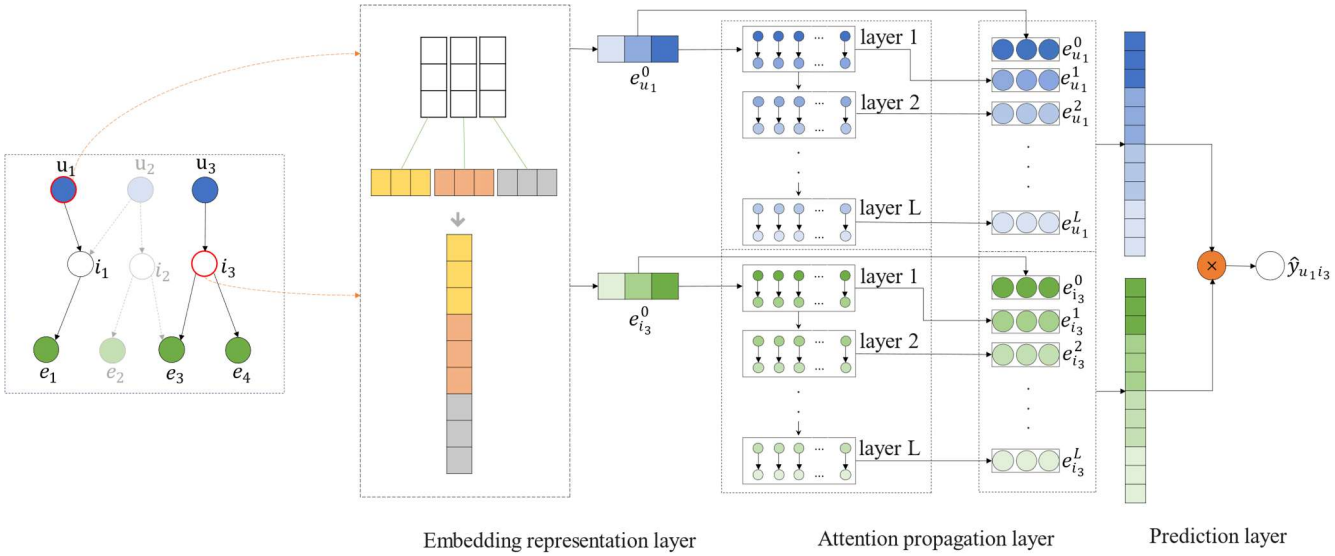


Fig. 1. The general structure of the proposed GATKR model consists of several primary modules, including the embedding layer, the attentive propagation layer, and the prediction layer.

obtain the corresponding prediction results.

A. Embedding Representation Layer

The process of knowledge graph embedding involves transforming entities and relations into vectors while preserving the structural features of KG. The complexity of different knowledge graph methods varies, and their suitability for various application scenarios also varies. Past work has heavily utilized trans-embedding methods, particularly TransR. However, this paper explores an alternative embedding method that differs from the translation distance model. We use the semantic matching model ConvKB for knowledge graph embedding.

ConvKB obtains triplet features through the convolutional layer filter. These features are then combined into a feature vector and weight vector, which are multiplied to obtain the prediction score. By continuously optimizing the weights, a suitable vector representation is obtained. Figure 3 illustrates the detailed steps for computing the triplet score in the embedding representation layer in Figure 1. The input matrix consists of e_h, e_r, $e_t \in \mathbb{R}^k$, $[\mathrm{e_h},\ \mathrm{e_r},\ \mathrm{e_t}] \in \mathbb{R}^{k\times 3}$, which are then processed by three 1×3 filters in the convolutional layer to extract global relations. The scoring function for each triplet is:

$$f(h,r,t) = concat\big(g([e_h,e_r,e_t]*\mu)\big)\cdot w \tag{1}$$

where $\mu\epsilon\mathbb{R}^{1\times 3}$ represents the filter, $w\epsilon\mathbb{R}^{|\mu|k\times 1}$ where $|\mu|$ denotes the number of μ, g stands for the non-linear activation function ReLU, and $concat$ denotes the concatenation operator that concatenates three vectors into one vector. As the value of $f(h,r,t)$ decreases, the probability that the triplet is true increases, and conversely, the probability is lower.

Valid and invalid triplets are trained in pairs and are distinguished by a loss function, which is as follows:

$$\mathcal{L}_{KG} = \sum_{(h,r,t)\epsilon G,(h,r,t'\epsilon G')} -\log\left(1+\exp\left(f(h,r,t')-f(h,r,t)\right)\right) \tag{2}$$

where G' replaces the tail entity of G, resulting in invalid triplets not existing in G, providing negative samples for embedding training.

B. Attention Propagation Layer

At the attention propagation layer, $e_{h'}$ is used as a substitute for the first-order features of the item consisting of the set of neighbors, with the following formula:

$$e_{h'} = \sum_{(h,r,t)\in h'} \pi(h,r,t)e_t \tag{3}$$

where $h' = \{(h,r,t)|(h,r,t)\in G\}$, t is the neighboring node of the head entity h, where e_t is the vector representation of the neighboring nodes. The attention weight $\pi(h,r,t)$ determines the weight of the neighboring information during message delivery. In Figure 2, the process of calculation of attentional weights in layer l and the process of attention propagation of item i_3 in Figure 1 in layer l are shown.

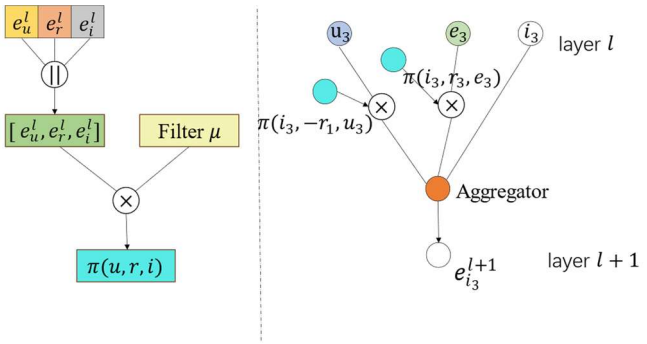


Fig. 2. The process of calculating attentional weights and attention propagation layer.

The formula for the attention weight $\pi(h,r,t)$ at the time of message delivery is as follows:

$$\pi(h,r,t) = \sigma(w*([e_h,e_r,e_t]*\mu)) \tag{4}$$

where σ is the non-linear activation function $tanh$, the filter applied to the vector splice of the triplets preserves the most important features, allowing the head entity to focus on the more important ones.

To prevent overfitting, $\pi(h,r,t)$ is normalized:

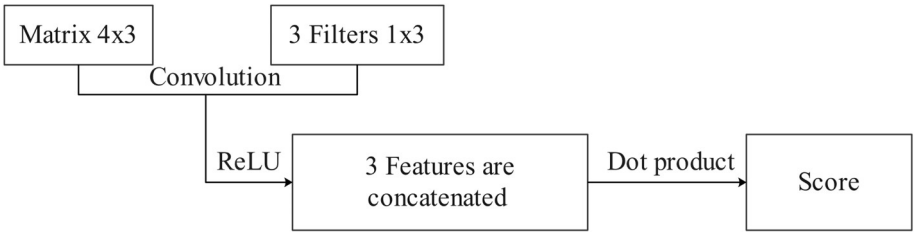


Fig. 3. Detailed steps for the embedding module ConvKB in the GATKR model.

$$\pi(h,r,t) = \frac{exp\ (\pi(h,r,t))}{\sum_{\pi(h,r',t')} exp\ (\pi(h,r,t))} \tag{5}$$

A layer of entities and first-order features from the Attention propagation layer can be used to expand to higher-order features. The subsequent layer of entity representations is obtained by using the entities and features from the previous layer as inputs:

$$e_h^l = g(e_h^{l-1}, e_{h'}^{l-1}) \tag{6}$$

where e_h^{l-1} and $e_{h'}^{l-1}$ represent the entity features and neighbor features, respectively. g is an aggregator that integrates the entity and neighbor features in a certain way, and the following aggregators are usually used:

(1) The GCN [17] aggregation, also known as summation aggregation, adds up the features of the entity and its neighbors and then applies a non-linear transformation. The formula is as follows:

$$g_{GCN} = \sigma(W(e_h + e_{h'}) + b) \tag{7}$$

where σ is the non-linear activation function $LeakyReLU$, $W \epsilon \mathbb{R}^{d \times d}$ is a trainable weight matrix, and $b \epsilon \mathbb{R}^d$ is the function bias.

(2) The GraphSage [18] aggregation, also known as connection aggregation, connects entities and their neighborhood features, followed by a non-linear transformation. The formula is as follows:

$$g_{GraphSage} = \sigma(W(e_h || e_{h'}) + b) \tag{8}$$

where $W \epsilon \mathbb{R}^{d \times 2d}$, $||$ is the connection operation.

(3) The Bi-Interaction [19] aggregation proposed in NFM is expressed as follows:

$$g_{Bi-Interaction} = \sigma(W_1(e_h + e_{h'}) + b_1) + \sigma(W_2(e_h \odot e_{h'}) + b_2) \tag{9}$$

where $W_1, W_2 \epsilon \mathbb{R}^{d \times d}$, $b_1, b_2 \epsilon \mathbb{R}^d$ are trainable weight functions with bias, and $\odot$ denotes vector dot product.

This layer captures higher-order features of the entity that help predict the results in the subsequent prediction layer.

C. Prediction Layer

Following the attention propagation layer, we obtain the higher-order features of both user and item nodes. Additionally, we need to acquire the initial representations of users and items, denoted as e_u^0, e_i^0 in Figure 1. These can be obtained from the entity-item comparison table (D) as outlined below:

$$e_i^0 = \{e_j | (i_j, e_j) \epsilon H, j = 1, \dots, N\} \tag{10}$$

$$e_u^0 = \left\{ \frac{\sum_{e_i^0} e_j}{|e_i^0|} \middle| (i_j, e_j) \epsilon H, (u,i) \epsilon R, j = 1, \dots, N \right\} \tag{11}$$

According to the above equation, the initial representation of e_i^0 is the representation e_j of the entity corresponding to the item in knowledge graph, where j is the corresponding number of i in D. The representation of e_u^0 is obtained by averaging the items e_i^0 that the user has interacted with in the user-item interaction graph G_1.

The final prediction result combines the higher-order features and merges the higher-order vector representations into a single vector:

$$e_u = e_u^0 || e_u^1 || \dots || e_u^L \tag{12}$$

$$e_i = e_i^0 || e_i^1 || \dots || e_i^L \tag{13}$$

where $||$ is the join operation.

Finally, the prediction score is obtained through the inner product:

$$\hat{y}_{ui} = e_u \top e_i \tag{14}$$

D. Model Optimization and Training

We treat non-existing interactions as negative samples to ensure the model's normal training process. This approach addresses the issue of insufficient training data, which can arise when all samples are positive. Our proposed loss function consists of three main parts in this paper:

$$\mathcal{L} = \mathcal{L}_{KG} + \mathcal{L}_{CF} + \lambda ||\Theta|| \tag{15}$$

where Θ is the set of parameters, including the weight function W, the embedding dimensions of users, items, entities, and so on, and the $\mathcal{L}_{CF}$ formula is:

$$\mathcal{L}_{CF} = \sum_{ui \epsilon R^+, uj \epsilon R^-} -ln\sigma(\hat{y}_{ui} - \hat{y}_{uj}) \tag{16}$$

where σ is the sigmoid function and R^+, R^- represent the presence and absence of interactions, respectively. $\mathcal{L}_{CF}$ is optimized using BPR loss [20], which can give higher prediction scores for observed interactions than unobserved interactions.

The use of the Adam [21] adaptive optimizer in this paper allows for the alternating optimization of L_{CF} and L_{KG}. This approach adaptively adjusts the learning rate and helps prevent overfitting during training. After updating the user and item representations layer by layer, the loss function updates

the model parameters. The optimal parameter that minimizes the loss is eventually found through continuous training.

V. Experiments

A. Dataset Description

In this paper, we use four types of datasets, Amazon-Book, Yelp2018, Last-FM and Alibaba, to verify the effectiveness of our algorithm. Amazon-Book is a subset of the Amazon dataset that mainly contains book-related data. Yelp2018 is a dataset about offline businesses, including restaurants, theatres, and so on, that treats businesses as items. Last-FM collects users' listening history in the Last-FM music app as data records, with songs viewed as items. Alibaba is an e-commerce dataset containing records of user and clothing interactions with clothing as items. We will filter users and items with more than 10 interaction records in order to guarantee the dataset's quality. Table I shows the details of the four datasets.

For each dataset, interactions are randomly partitioned into training (80%), testing (20%), and validation (10% subset of training) sets to ensure robust evaluation.

TABLE I
Statistics of the experimental datasets.

	Amazon-Book	Yelp2018	Last-FM	Alibaba
Interactions				
Users	70,679	45,919	23,566	114,737
Items	24,915	45,538	48,123	30,040
Interactions	847,733	1,185,068	3,034,796	1,781,093
Knowledge Graph				
Entities	88,572	90,961	58,266	59,156
Relations	39	42	9	51
Triplets	2,557,746	1,853,704	464,567	279,155

B. Evaluation Indicators

We use recall and ndcg to measure the effectiveness and performance of our model. Recall is obtained by the ratio of the number of items in the test set that the user interacted with and the number of items that the model predicted correctly, with the formula:

$$recall = \frac{N_{TP}}{N_{TP}+N_{FN}} \tag{17}$$

where N_{TP} is the number of items recommended by the model and user interacted with, and N_{FN} is the number of items not recommended by the model but with which the user interacted.

ndcg is used to assess the model's ranking performance using the formula.:

$$ndcg = \frac{dcg}{idcg} \tag{18}$$

where dcg is the discounted cumulative gain, which calculates the score of the list of items recommended by the model, the score is impacted by the rank of the items, with higher ranked items having a more significant impact on the score, with the formula:

$$dcg = \sum_{i=1}^{k} \frac{rel(i)}{log_2(i+1)} \tag{19}$$

where $rel(\cdot)$ represents the relevance score of the item, while $idcg$ stands for the ideal dcg, which is the dcg score when optimal alignment is achieved.

C. Baseline

To validate the effectiveness of our model, we compared it to seven popular recommendation models, divided into the following categories: traditional model (MF), embedding-based models (CKE, CFKG), and graph neural network-based models (RippleNet, KGCN, KGNN-LS, CKAN, KGAT), as follows:

MF [22]: As a standard matrix decomposition model, it does not incorporate a knowledge graph. It solely relies on the user-item interaction graph to make predictions through the decomposition matrix of users and items.

CKE [9]: This is the classic embedding-based recommendation model. It learns vector representations of users and items and makes recommendations based on these vector representations. The recommended results are generated through joint training in recommendation and knowledge graph.

CFKG [23]: This model integrates heterogeneous knowledge from multiple sources and models the entities and relationships of the knowledge graph in a unified vector space where recommendations are made by similarity.

RippleNet [24]: This model combines the embedding-based method and the path-based method. It uses the "water waves "-like way to propagate from the user's items of interest and spread over the knowledge graph to obtain the user's potential preferences and make recommendations.

KGCN [14]: The model leverages users and relationships to capture potential user interests.

KGNN-LS [25]: This model applies graph neural networks to the knowledge graph scoring function using user-specific relationships. It then aggregates it with neighborhood information of varying weights to learn edge weights.

CKAN [26]: This model uses an asynchronous propagation strategy to encode collaborative signals explicitly. It also employs knowledge-aware attention mechanisms to distinguish between different neighbors.

KGAT [6]: This model utilizes an attention mechanism for end-to-end modeling based on higher-order connectivity. It is also the first model to merge the user-item interaction graph and knowledge graph into a collaborative knowledge graph for training.

D. Experimental Settings

The code presented in this paper is based on Pytorch and employs the Xavier initializer to initialize the model parameters. The learning rate is adjusted among $\{5\times10^{-2}, 1\times10^{-2}, 5\times10^{-3}, 1\times10^{-3}\}$, the L_2 regularization coefficients are adjusted among$\{1\times10^{-5}, 1\times10^{-4}, \ldots, 1\times10^{1}, 1\times10^{2}\}$, and the dropout rate is adjusted among {0.0,0.1,0.2, …,0.8}. The embedding dimensions for the user, item, entity, and relationship are all set to 64. RippleNet is set to 16 due to computational costs and the batch size is fixed at 1024. In the interest of minimizing unnecessary expenditure of computational resources, an early-stopping strategy has been utilized. Training is stopped when there is no improvem-

TABLE II
PERFORMANCE COMPARISON ON AMAZON-BOOK, YELP-2018, LAST-FM, AND ALIBABA DATASETS IN TERMS OF RECALL@20 AND NDCG@20.

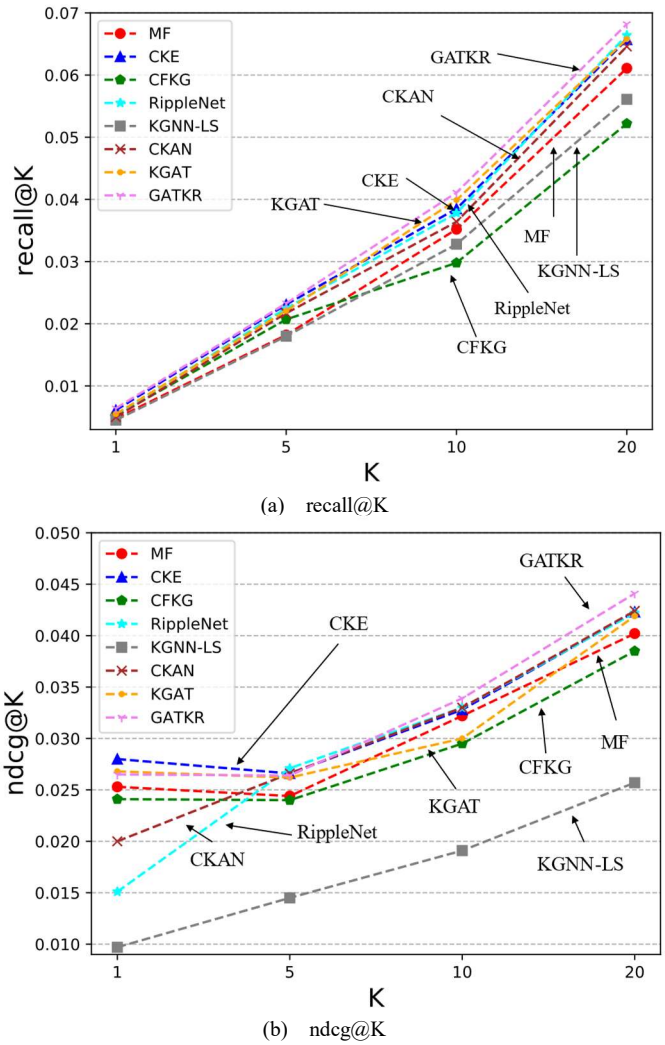
	Amazon-Book		Yelp2018		Last-FM		Alibaba	
	recall	ndcg	recall	ndcg	recall	ndcg	recall	ndcg
MF	0.1300	0.0678	0.0618	0.0402	0.0724	0.0617	0.0506	0.0276
CKE	0.1342	0.0698	0.0657	0.0423	0.0732	0.0630	0.0835	0.0512
CFKG	0.1264	0.0663	0.0522	0.0395	0.0723	0.0613	0.0901	0.0516
RippleNet	0.1336	0.0697	0.0664	0.0422	0.0791	0.0655	0.0960	0.0521
KGCN	0.1398	0.0569	0.0532	0.0338	0.0839	0.0694	0.0983	0.0633
KGNN-LS	0.1362	0.0560	0.0561	0.0257	0.0840	0.0642	0.1039	0.0633
CKAN	0.1380	0.0698	0.0646	0.0424	0.0812	0.0660	0.0970	0.0509
KGAT	**0.1404**	**0.0746**	0.0658	0.0419	0.0819	0.0701	0.1030	0.0627
GATKR	0.1391	0.0728	**0.0682***	**0.0441***	**0.0847***	**0.0721***	**0.1066***	**0.0653***
%Improv.	-0.9%	-2%	2.7%	4%	0.8%	2%	2.6%	3.1%

ent in the recall@20 for 10 rounds of the validation set.

E. Experimental Results and Analysis

Table II presents the effectiveness of recommendation models in the Amazon-Book, Yelp2018, Last-FM and Alibaba datasets. The evaluation measures that are employed are ndcg@20 and recall@20.

The best data in the baseline model are underlined, while the best data containing our proposed algorithmic model is presented in bold. The %Improv. represents the gap between the proposed GATKR model and the best data in the baseline. Figure 4 presents a comparison of model performance for recall@K and ndcg@K at various K values using the Yelp 2018 dataset.



(a) recall@K

(b) ndcg@K

Fig. 4. Comparison of recall and ndcg in Top-K recommendations in the Yelp2018 dataset.

From the experimental results in Table II and Figure 4, the following conclusions can be drawn:

(1) GATKR demonstrates optimal performance on the Yelp2018, Last-FM, and Alibaba datasets, exhibiting substantial enhancements in both metrics. Specifically, on the Yelp2018 dataset, GATKR enhances recall@20 by 2.7% and ndcg@20 by 4% compared to the optimal baseline model. The Last-FM dataset shows a 0.8% improvement in recall@20 and a 2% improvement in ndcg@20, and the Alibaba dataset demonstrates a 2.6% enhancement in recall@20 and a 3.1% improvement in ndcg@20.

(2) On the Amazon-Book dataset, a performance gap exists between GATKR and KGAT, with Recall@20 and NDCG@20 decreasing by 0.9% and 2%, respectively. These results suggest that our knowledge graph embedding method may diminish in effectiveness as the number of triplets increases significantly. Additionally, entity representation is biased when the number of neighbors is too large for attention propagation, causing the filter to exclude important neighbors.

(3) In top-k recommendation, the recall improves significantly with increasing k in Yelp2018 dataset. The performance of GATKR compared with KGAT proves the effectiveness of the filter in filtering out useless information. The effectiveness of graph attention in aggregating neighbor information is demonstrated by the results when compared to KGNN-LS.

(4) The performance of KGNN-LS in Figure 4(a) is comparable to that of the other models, yet it experiences a notable decline in Figure 4(b). This suggests that the Yelp2018 dataset may have a higher degree of irrelevant information, which KGNN-LS is unable to filter out effectively. Consequently, this irrelevant information negatively impacts the recommendation performance.

(5) CKE and CFKG are also embedding-based models. Notably, CFKG exhibits poorer performance compared to MF when the knowledge graph is not utilized effectively. This observation suggests that inadequate integration of the knowledge graph in embedding-based models can negatively impact recommendation performance.

(6) RippleNet is a path-based model. In the implementation method, interest diffusion is similar to the message passing of GNN. Therefore, RippleNet performs similarly to models based on GNN.

(7) GATKR improves the most on the Yelp2018 and Alibaba datasets, and the statistics from the two datasets show that their knowledge graphs have more complex relationships, and GATKR's embedding module is better at handling complex relationships and thus has better performance.

(8) The superior performance of models based on graph neural networks (KGCN, KGNN-LS, CKAN, KGAT) compared to other models across four datasets suggests the significance of graph neural networks messaging mechanism in enhancing knowledge graph-based recommendations.

F. Ablation Experiment

In order to assess the impact of various components in GATKR on the recommendation effect, an experimental design was developed for comparison. The experiments were carried out by varying the number of graph attention network layers, using different aggregation functions, and deciding whether to use knowledge graph embedding and graph attention network, respectively.

Number of Graph Attention Network Layers. The number of layers in the GAT was adjusted in the range of {1,2,3,4}. Table III shows the effect of different layers on the recommendation.

(1) As the number of layers in the graph attention network increases, the quality of recommendations improves, particularly when the number of layers is 2 or 3. However, when the layer count reaches 4, the effectiveness of the recommendations begins to decline, exhibiting a performance that is even lower than that of a network with only 1 layer. This observation indicates that the augmentation of layers in a graph attention network may potentially result in the loss of salient features, thereby compromising the efficacy of the model.

(2) Upon jointly analyzing Tables II and III, it is evident that the GATKR recommendation outperforms most baseline models even when the number of layers in the GAT is 1.

TABLE III
EFFECT OF DIFFERENT GRAPH ATTENTION NETWORK LAYERS.

Number of GAT layers	Amazon-Book		Yelp2018		Last-FM	
	recall	ndcg	recall	ndcg	recall	ndcg
1	0.1387	0.0729	0.0677	0.0438	0.0837	0.0710
2	0.1389	**0.0735**	**0.0687**	**0.0444**	0.0841	0.0716
3	**0.1391**	0.0728	0.0682	0.0441	**0.0847**	**0.0721**
4	0.1384	0.0729	0.0664	0.0426	0.0827	0.0706

Aggregation Functions in Attention Propagation Layer. Comparative experiments were conducted on three aggregators, GCN, GraphSage, and Bi-Interaction, with a single layer of GAT. The effects of selecting different aggregators are documented in Table IV.

The Bi-Interaction aggregator has the best recommendation because it incorporates multimodal information. In contrast, the GCN and GraphSage aggregators aggregate information in a single way, resulting in the loss of information from other aspects.

TABLE IV
EFFECT OF DIFFERENT AGGREGATORS.

Aggregators	Amazon-Book		Yelp2018		Last-FM	
	recall	ndcg	recall	ndcg	recall	ndcg
GCN	0.1358	0.0712	0.0669	0.0431	0.0825	0.0697
GraphSage	0.1358	0.0713	0.0670	0.0432	0.0834	0.0707
Bi-Interaction	**0.1387**	**0.0729**	**0.0677**	**0.0438**	**0.0837**	**0.0710**

Knowledge Graph Embedding and Graph Attention Network. To evaluate the effectiveness of the knowledge graph embeddings and graph attention networks we employed, we designed three cases: disabling the knowledge graph embeddings, disabling the graph attention networks, and disabling both simultaneously. The experimental results are presented in Table V.

(1) The impact on recommendation performance is more pronounced when the graph attention network is disabled compared to when the knowledge graph embedding is disabled. This indicates the critical importance of enriching entity information with neighborhood information. Without the GAT, the model proposed in this paper reverts to a standard knowledge embedding-based recommendation model.

(2) If knowledge graph embedding and GAT are disabled, the model proposed in this paper becomes the same as the traditional recommendation model. This model is unable to process auxiliary information and can only make recommendations based on historical interactions, resulting in a further degradation of recommendation performance.

TABLE V
IMPACT OF KNOWLEDGE GRAPH EMBEDDING AND GAT.

Disable Components	Amazon-Book		Yelp2018		Last-FM	
	recall	ndcg	recall	ndcg	recall	ndcg
Embedding	0.1373	0.0715	0.0664	0.0420	0.0825	0.0695
GAT	0.1370	0.0712	0.0662	0.0419	0.0823	0.0693
E&G	0.1364	0.0706	0.0658	0.0412	0.0818	0.0688

G. Further Investigation on GATKR

Knowledge Graph Noise. To evaluate GATKR's filtering ability in the face of noise, we add noise triples to the data and compare it with existing knowledge graph recommendation models based on GNN. We tested the performance decrease of the model under the same measurement set by adding 10% of noise triples to the KG. The experimental results are shown in Figure 5.

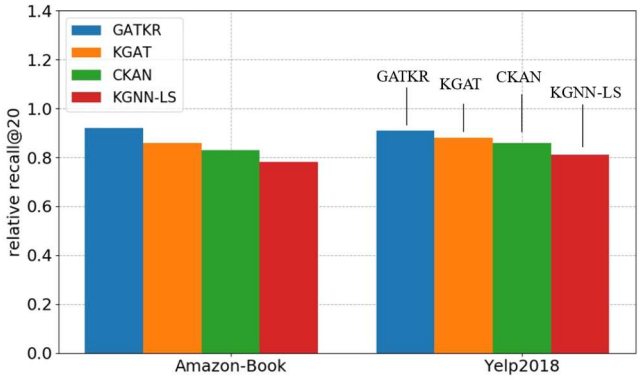
Sparse Item Interaction Data. To evaluate the recommendation ability of GATKR in long-tail items, we divided all the items into five groups according to the interaction density, and the interaction density increased in order. The experimental results are shown in Figure 6.

(1) Compared to knowledge graph recommendation models that rely on graph neural networks, GATKR demonstrates robust performance in the presence of noise in the knowledge graph. This robustness is primarily attributed to the filter incorporated in the GAT, which effectively eliminates invalid triples. As shown in Figure 5, the recall and ndcg performance of GATKR decreases the least in both datasets.

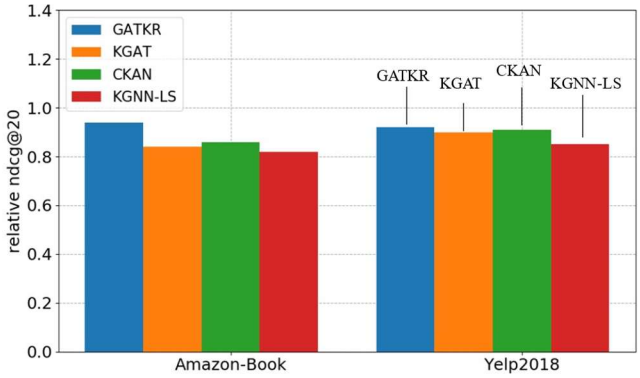
(2) The sparsity of item interaction data frequently gives rise to long-tail effects. GATKR, however, has been shown to exhibit superior resilience in such contexts. In contrast, KGAT and CKAN exhibit a heightened susceptibility to interference from irrelevant interactions when confronted with sparse data, resulting in recommendations that are subject to bias. This bias can be effectively mitigated by the high-quality vector representations extracted through our model.

H. Case Study

Our case study on Amazon Book, which is seen in Figure 7, demonstrates how noise filtering and graph attention propagation work. We randomly select user u_{200} and item i_{631} (unseen in the training set) for the demonstration. u_{200}

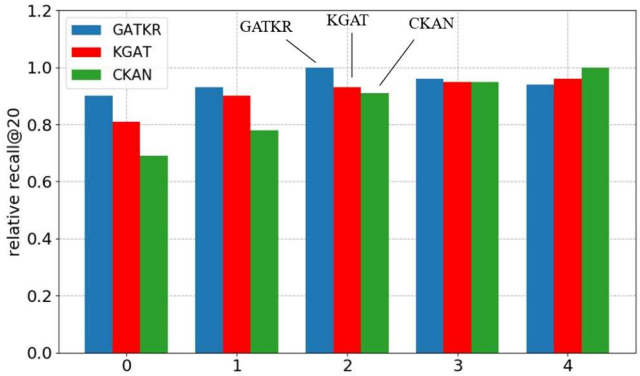


(a) relative recall

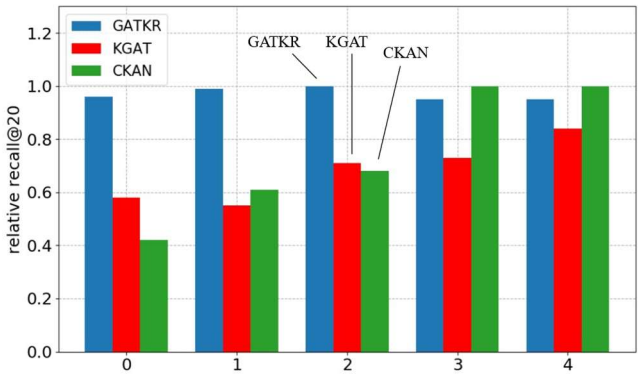


(b) relative ndcg

Fig. 5. Performance in alleviating KG noise.



(a) Amazon-Book



(b)Yelp2018

Fig. 6. Performance in different data sparsity degrees.

has interacted with *Harry Potter* and *Pride and Prejudice*, and the knowledge graph provides relevant information about these two books, not all of which are useful. By filtering the unnecessary information, we can get the path to recommend i_{631} to u_{200}, it has the highest attention score and is the solid line in Figure 7. We know that the reason for recommending *The Lord of the Rings* to u_{200} is that the user has read *Harry Potter*, which is also a magic book, and the user prefers magic books.

VI. Conclusion

This paper proposes a filter-based attention mechanism knowledge graph recommendation algorithm (GATKR). The algorithm consists of three layers: prediction, attention propagation and embedding representation. The attention propagation layer filters the features obtained from GAT using filters to pay more attention to the features that account for a large proportion and capture the entity's main features. Comparative experiments on four datasets and ablation experiments demonstrate the effectiveness of our proposed algorithm. The algorithm successfully improves the representation of entity features on GAT. In addition, the algorithm effectively mitigates the effect of knowledge graph noise.

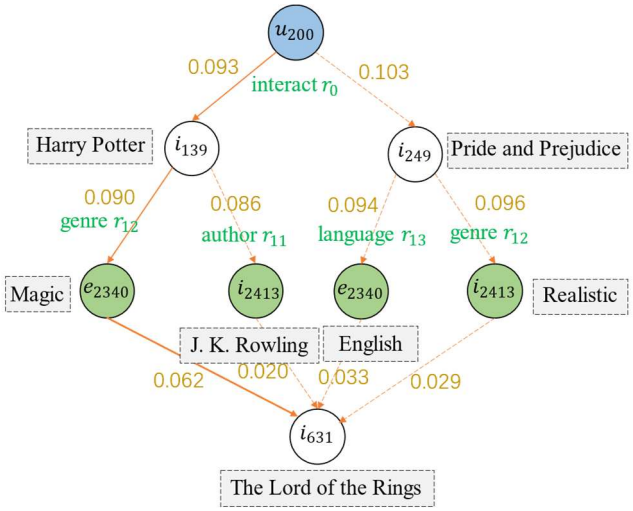


Fig. 7. Real example from Amazon-Book dataset.

Reference

[1] X. He, L. Liao, H. Zhang, L. Nie, and T.S. Chua, "Neural collaborative filtering" in World Wide Web Conferences, Perth, Australia, pp. 173-182, 2017.

[2] X. Wang, X. He, M. Wang, F. Feng, T.S. Chua, "Neural Graph Collaborative Filtering" in Special Interest Group on Information Retrieval, Paris, France, 2019.

[3] Y. Cao, H. Lei, J. Li, and Z. Liu, "Neural Collective Entity Linking" in COLING, Santa Fe, USA, pp. 675–686, 2018.

[4] Y. Cao, L. Hou, J. Li, Z. Liu, and T. Dong, "Joint Representation Learning of Cross-lingual Words and Entities via Attentive Distant Supervision" in Empirical Methods in Natural Language Processing, Brussels, Belgium, pp. 227–237, 2018.

[5] C. Huang, and Y. Zhong, "A Novel Approach for Representing Temporal Knowledge Graphs," IAENG International Journal of Computer Science, vol. 51, no. 6, pp. 694-702, 2024.

[6] X. Wang, X.N. He, Y.X. Cao, M. Liu, and T.S. Chua, "KGAT: Knowledge Graph Attention Network for Recommendation" in Knowledge Discovery and Data Mining. San Francisco, USA, pp. 950–958, 2019.

[7] DQ. Nguyen, T.D. Nguyen, D.Q. Nguyen, and D. Phung, "A Novel Embedding Model for Knowledge Base Completion Based on Convolutional Neural Network," in 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, New Orleans, pp.327-333, 2018.

[8] Q. Wang, and W. Zhang, "Session-based Recommendation Algorithm Based on Heterogeneous Graph Transformer," IAENG International Journal of Computer Science, vol. 50, no.4, pp. 1347-1353, 2023.
[9] F. Zhang, N.J. Yuan, D. Lian, X. Xie, and WY Ma, "Collaborative Knowledge Base Embedding for Recommender Systems" in Knowledge Discovery and Data Mining, San Francisco, USA, pp. 353–362, 2016.
[10] Y.K. Lin, Z.Y. Liu, MS Sun, Y. Liu, and X. Zhu, "Learning Entity and Relation Embeddings for Knowledge Graph Completion" in American Association for Artificial Intelligence, California, USA, pp. 2181–2187, 2015.
[11] H. Wang, F. Zhang, and M. Zhao, "Multi-task feature learning for knowledge graph enhanced recommendation" in World Wide Web Conferences, San Francisco, USA, pp. 2000-2010, 2019.
[12] K.Y.L. Xu, C.T. Li, YL Tian, and T. Sonobe, "Representation Learning on Graphs with Jumping Knowledge Networks" in International Conference on Machine Learning, Stockholm, Sweden, Vol. 80, pp. 5449–5458, 2018.
[13] Y. Teng, and K. Yang, "Research on Enhanced Multi-head Self-Attention Social Recommendation Algorithm Based on Graph Neural Network" IAENG International Journal of Computer Science, vol. 51, no. 7, pp. 754-764, 2024.
[14] H. Wang, M. Zhao, X. Xie, W. Li, and M. Guo, "Knowledge graph convolutional networks for recommender systems", https://doi.org/10.48550/ arXiv.1412.6980, 2019.
[15] X. Sha, Z. Sun, and J. Zhang, "Attentive Knowledge Graph Embedding for Personalised Recommendation", https://doi.org/10.48550/arXiv.1811.10776, 2019.
[16] Y. X. Geng, L. Wang, Z. Y. Wang and Y. G. Wang, "Central Attention Mechanism for Convolutional Neural Networks," IAENG International Journal of Computer Science, vol. 51, no. 10, pp. 1642-1648, 2024.
[17] N. Thomas, Kipf, and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks" in International Conference on Learning Representations. Toulon, France, 2017.
[18] W.L. Hamilton, R. Ying, and J. Leskovec, "Inductive Representation Learning on Large Graphs". in Conference and Workshop on Neural Information Processing Systems, pp. 1025–1035, 2017.
[19] He X., and Chua T.S., (2017), "Neural Factorization Machines for Sparse Predictive Analytics" in Special Interest Group on Information Retrieval, Tokyo, Japan, pp. 355-364, 2017.
[20] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian Personalized Ranking from Implicit Feedback" in Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, Canada, June 18-21, pp. 452–461, 2009.
[21] PK Diederik, and B. Jimmy, "Adam: A Method for Stochastic Optimization", https://doi.org/10.48550/arXiv.1811.10776, 2014.
[22] Y. Koren, R. Bell, and C. Volinsky, "MATRIX FACTORIZATION TECHNIQUES FOR RECOMMENDER SYSTEMS" in IEEE, pp.30-37, 2009.
[23] QY Ai, V. Azizi, X. Chen, and Y.F. Zhang, "Learning Heterogeneous Knowledge Base Embeddings for Explainable Recommendation". Algorithms, Vol.11 No.9, pp. 137, 2018.
[24] H. Wang, F. Zhang, J. Wang, M. Zhao, and M. Guo, "RippleNet: Propagating User Preferences on the Knowledge Graph for Recommender Systems" in Conference on Information and Knowledge Management, Turin, Italy, pp. 417–426, 2018.
[25] H. Wang, F. Zhang, M. Zhang, and J. Leskovec, "Knowledge-aware Graph Neural Networks with Label Smoothness Regularization for Recommender Systems" in Knowledge Discovery and Data Mining, San Francisco, USA, pp. 968–977, 2019.
[26] Z. Wang, G.Y. Lin, H.B. Tan, Q.H. Chen, and X.Y. Liu, "CKAN: Collaborative Knowledge-aware Attentive Network for Recommender Systems" in Special Interest Group on Information Retrieval, Xi" an China, pp. 219–228, 2020.