Neural Processing Letters
https://doi.org/10.1007/s11063-022-11077-0

Knowledge Graph Attention Network with Attribute Significance for Personalized Recommendation

Chenyu Wang[1] · Haiyang Zhang[1] · Lingxiao Li[1] · Dun Li[1]

Accepted: 18 October 2022

Abstract

The recommendation system based on the knowledge graph usually introduces attribute information as supplements to improve the accuracy. However, most existing methods usually treat the influence of attribute information as consistent. To alleviate this problem, we propose a personalized recommendation model based on the attribute significance of the knowledge graph attention network (AS-KGAN). Firstly, to obtain user preferences on the knowledge graph and maintain the inherent structure and semantic information, we consider the entity types in the knowledge graph and conduct modeling in an end-to-end approach. Secondly, when introducing item attribute information, we consider the significance of different attributes on items and use the graph attention network to distinguish them. Thirdly, to obtain deeper representations of user preferences, the weights of node neighborhoods are learned during propagation. Finally, to better obtain the high-order relationships and ensure interpretability, we model the high-order connectivity explicitly in the knowledge graph. The model generates *top-K* recommendations for target users by predicting the probability that users will interact with the item. We applied this model to three public datasets, and the experimental results indicate that the recommendation quality of our model has improved compared with other models.

Keywords Knowledge graph · Recommender systems · Graph attention networks · Attribute significance

✉ Dun Li
ielidun@zzu.edu.cn

Chenyu Wang
iewangchenyu@163.com

Haiyang Zhang
1713911965@qq.com

Lingxiao Li
lilingxiao6438@gmail.com

[1] School of Computer and Artificial Intelligence, Zhengzhou University, No.100 Science Avenue, Zhengzhou 450001, Henan, China

1 Introduction

With the development and progress of mobile Internet in recent years, people can obtain and check information more conveniently, which has brought great convenience to users. In the meantime, the number of users and data information rapid growth on the network have led to the phenomenon of information overload. One of the effective ways to solve the problem of information overload is personalized recommendation [1]. With the growing number of users and items, the recommendation accuracy of traditional collaborative filtering algorithms continues to decline due to problems such as cold start, sparse data, and less consideration of the influence of historical project preferences. To solve these problems, researchers have tried to introduce additional information into the recommendation model to enrich the description of users, items, or information, thereby making up for the sparsity or deficiency of users' historical preference datasets. Commonly used auxiliary information includes the social network [2], user/item attributes [3], image/text multimedia information [4], context information [5], knowledge graph, etc.

The knowledge graph [6] is a kind of directed heterogeneous graph. In the knowledge graph, nodes represent entities, and edges represent semantic relationships between entities. The advantages of introducing the knowledge graph into the recommendation system are as follows: (1) Knowledge graph can not only provide richer semantic association between users and items but also provide diversified auxiliary information sources for the recommendation system. It can dig deeper into relationships between items, improve the recommendation accuracy, and effectively relieve the sparseness and cold start problems of user behavior records; (2) The implicit information obtained by reasoning provides personalized services for users and is widely used in decision support, electronic commerce, natural language understanding, text classification. The knowledge graph contains a variety of relationships, which can reasonably expand users' interest range from multiple directions. This makes the recommendation system avoid too single recommendation results and enhances the diversity of recommended items; (3) The knowledge graph connects users' preferred items with the recommended items, which improves the accuracy and interpretability of recommendation results, and thus improves users' satisfaction and acceptance of the recommendation results.

When using the knowledge graph for the recommendation, the data information is first collected according to the recommended task features. Data information generally includes user information, item information, and user-item interaction information (For instance, user's clicks, purchases, collection and other historical behaviors or user's ratings, etc.). Knowledge graphs used for recommendation are constructed based on the collected data (or linking external data). Then the recommendation model uses the collected data and the constructed knowledge graph to train and verify. And the trained recommendation model generates recommendations for users.

In the research of recommendation systems based on the knowledge graph, attribute information of users or items is usually introduced as supplementary to make personalized recommendations to target users. Taking movie recommendation as an example, user attributes include the user's gender, age, occupation, etc., and movie attributes include starring, director, type, etc. Existing research generally treats the item attribute significance as the same. However, when exploring users' interests, different attributes of the same item have different influences on users. A sample knowledge graph as shown in Fig. 1. Both user 1 and user 2 like movie A, user 1 also likes another movie C with the same starring in movie A, and user 2 also likes movie B of the same type as movie A. When obtaining the preferences of user 1 and user 2, it is unreasonable to set the significance of movie A's attributes on users

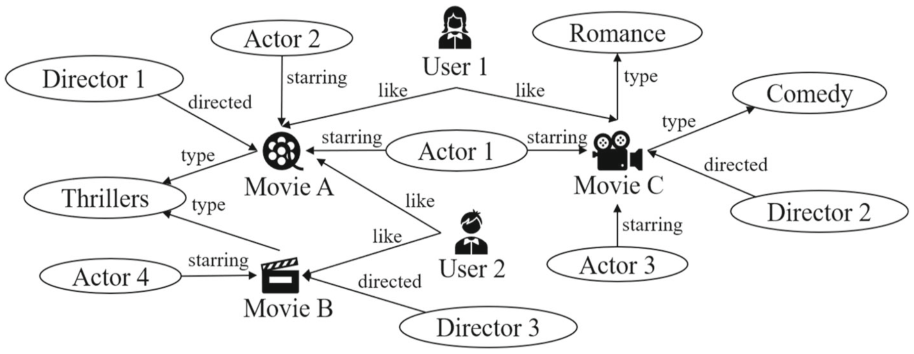

Fig. 1 Sample of knowledge graph

as the same. In the knowledge graph, the types of entities are different. In the knowledge graph in Fig. 1, the entities include users, movie names, actors and directors, etc. Users and movie names represent people and things respectively and belong to different types of entities. Therefore, it is inappropriate to process different entities in the same way.

To alleviate these problems, we propose a personalized recommendation model based on the attribute significance of the knowledge graph attention network (AS-KGAN). The diversity of entities and relationships in the knowledge graph is considered when making recommendations based on the knowledge graph. The method defines two vectors for each entity and relationship, which makes each entity-relationship pair have a unique mapping matrix. To improve the recommendation accuracy, the model introduces item attributes as supplementary information for the knowledge graph, and the influence of different attributes of items on users is considered. The main content of the model includes:

(1) We consider the types of entities and relationships simultaneously when making recommendations based on the knowledge graph. The data is processed by the knowledge representation learning method. This method maintains the inherent structure and semantic information of the knowledge graph.
(2) We consider the significance of different item attribute information when introduced into the model as supplementary information of the knowledge graph. The model uses the graph attention network to distinguish the significance of item attributes and learns the weights of node neighborhoods in the propagation process to obtain a deeper representation of user preferences.
(3) The model generates a recommendation list for the target user by predicting the probability that the user will interact with the item. The proposed model is compared with the existing recommended benchmark methods on the datasets MovieLens-20M, Amazon-book, and Last.FM.

In this paper, Sect. 2 describes the work of recommendation by using the knowledge graph; Sect. 3 describes the personalized recommendation model based on attribute significance of the knowledge graph attention network; Sect. 4 describes the overall scheme design of the experimental and performance comparison to verify the validity of model; Sect. 5 summarizes and discusses the further research work.

2 Related Works

The existing recommendation models based on the knowledge graph mainly include path-based methods, knowledge graph embedding methods, and propagation-based methods.

The path-based method mainly predicts the model by predefining the meta-paths and then using them as input. The network structure of the knowledge graph is fully and intuitively utilized. For example, Yu et al. [7] proposed the PER model. The connections between users and items are represented by extracting potential characteristics based on meta-paths or meta-graphs through the kinds of relational paths. Hu et al. [8] proposed the model MCRec. This is a three-way neural interaction model, which explicitly incorporates meta-path-based contextual design. In the model, prominent path instances are selected by using priority-based sampling techniques. Zhao et al. [9] proposed a meta-graph model to construct features between items based on meta-paths or meta-graphs. To improve the interpretability of path-based recommendations, Wang et al. [10] proposed the KPRN model. The model uses weighted pooling to distinguish the importance of different paths for prediction. To further explore the intent of user-item interactions, Wang et al. [11] proposed the KGIN model. They designed a new information aggregation method to recursively integrate the relational paths of remote connections.

Path-based methods rely on artificially designed meta-paths or meta-graphs. When designing meta-paths or meta-graphs, domain knowledge is often required, which makes model optimization difficult in reality and impractical for large-scale knowledge graphs. To deal with this problem, some researchers proposed a method based on knowledge graph embedding [12] (KGE). For example, Zhang et al. [13] proposed the CKE model. Firstly, the model learns the knowledge graph features, text features, and image features of items. Then use the CKE framework to jointly learn the item representation in collaborative filtering and the semantic representation in the knowledge base. Wang et al. [14] proposed the DKE model. The feature vector of the news entity is obtained by fusing the vectors through the framework of the convolutional neural network. Nathani et al. [15] proposed an attention-based feature embedding model, which encapsulates relationship clustering and multi-hop relationship in the model. Zhao et al. [16] proposed the UGRec model to construct a heterogeneous graph containing both directed knowledge relations and undirected item-item co-occurrence relations to model user preferences.

The method based on the knowledge graph embedding avoids preprocessing the knowledge graph and defining a large number of meta-paths. However, these methods are usually modeled in an implicit way and do not fully explore the potential preferences of users. Combining the above two methods, some researchers have proposed a propagation-based method. The recommendation performance is enhanced by iteratively performing information dissemination across the entire knowledge graph. For example, Wang et al. [17] proposed the RippleNet model. It constantly and automatically discovers users' latent hierarchical interests along the relationships in the knowledge graph, and effectively improves the limitations of recommendation methods based on path and knowledge graph embedding. To combine the potential key collaboration information and knowledge information in user-project interaction, Wang et al. [18] proposed the CKAN model to encode these two kinds of information by adopting the heterogeneous propagation strategy. However, in the RippleNet model, the importance of the relationship is weakly characterized. The increasing size of the knowledge graph will bring a lot of computing and storage overhead.

In recent years, some researchers have proposed a method to process graph data based on the graph neural network [19]. In the neural network, graph neural networks can be directly used for graph structure. It has a strong capability to model the dependency relationship between nodes in the graph [20]. For example, Hamilton et al. [19] proposed the GraphSage algorithm. GraphSage obtains local neighbor aggregation feature information of vertices by training a group of aggregator functions. Berg et al. [21] proposed the GC-MC model. In this model, graph neural networks are applied to matrix completeness problems with structured edge information. Wang et al. [22] proposed the KGCN model, which is based on the knowledge graph convolutional network. The heterogeneous information of the knowledge graph is used to aggregate the attributes of items and other additional auxiliary information, which enriches the representation of the item. However, it does not consider the user's attribute information, which leads to some defects in the recommendation model. Li et al. [23] proposed the DEKGCN model to improve the KGCN model by adding auxiliary information for users.

Through the study of the above methods, we found that when researchers use graph neural networks to implement the knowledge graph-based recommendation system, they usually do not consider the impact of item-side attributes in knowledge graphs on users' implicit feedback. To alleviate this problem, we propose a personalized recommendation model based on the attribute significance of the knowledge graph attention network. When the model introduces the attribute information of the item in the knowledge graph as auxiliary information, it distinguishes the influence of different attribute information on the item and considers the entity type in the knowledge graph. This method performs modeling in an end-to-end manner and better captures the high-order relationships in the overall knowledge graph. The high-order information is effectively utilized to ensure the interpretability of the model. At the same time, the attention mechanism is used to automatically learn the weights of different attributes of the item, and distinguish the importance of different attributes to construct preferences for users' implicit feedback. This exhibits the user's personal interest and improves the performance of recommendations.

3 Knowledge Graph Attention Network with Attribute Significance

In this section, we introduce the proposed Knowledge Graph Attention Network with Attribute Significance model (AS-KGAN). Fig. 2. shows the overall framework of the model.

The model is composed of three parts: (1) Pretreatment layer. The entity and relationship are parameterized into vector representation by the knowledge graph embedding method. (2) Graph attention embedding layer. Neighbor representations of nodes embed updates through recursive propagation. In the propagation process, the weight of item attributes is learned through the graph attention mechanism. (3) Prediction layer. This layer aggregates the user representation and item representation of all propagation layers and outputs the predicted matching score. We list the key symbols used in this article in Table 1.

3.1 Problem Formulation

The knowledge graph is a kind of directed heterogeneous graph. In the graph, nodes represent entities and edges represent semantic relationships between entities. The knowledge graph is defined as Eq. 1:

$$G = \{(h, r, t) | h, t \in E, r \in R\} \tag{1}$$

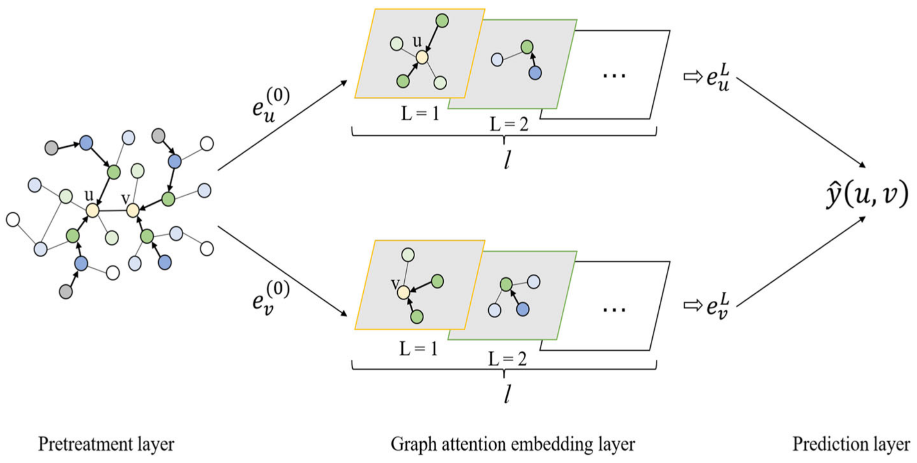

Fig. 2 Framework of AS-KGAN

Table 1 Key symbols used in this article

Notation	Description
$G = (h, r, t)$	Knowledge graph
U={u_1,u_2,...,u_M}	Set of users
V={v_1,v_2,...,v_N}	Set of items
y_{uv}	User-item interaction
e_h, e_t, e_r	Embedding for h, r, t
$\mathcal{E}_h$	Projected representation of e_h
$\mathcal{E}_t$	Projected representation of e_t
M_{rh}, M_{rt}	Mapping matrix
h_p, t_p	Entity semantic vector
r_p	Relational semantic vector
$I^{m\times n}$	Identity matrix
$h^t_{N_h}$	Neighbor structure of entity h

Where (h,r,t) represent the triple's head entity, relationship, and tail entity respectively. E and R represent the entity set and relationship set in the knowledge graph respectively. The recommendation problem of the knowledge graph is expressed as follows: In the recommendation scenario, there is usually a group of user set U and a group of item set V, represented as U={u_1,u_2,...,u_M} and V={v_1,v_2,...,v_N} respectively, where M represents the number of users, N represents the number of items. Y={(u,y_{uv},v)|$u \in U$,$v \in V$} represents the interaction between the user and the item. When y_{uv}=1, it means there is an interaction between the user and the item (such as reading, collecting), otherwise y_{uv}=0. In the sample knowledge graph in Fig. 1, for Movie A there are triples (*Movie A, is directed by, Director 1*) and (*User 2, like, Movie A*), where *Movie A*, *Director 1* and *User 2* represent entity sets, and *is directed by* and *like* represent relation sets. In the triple (*User 2, like, Movie A*), *User 2* interacts with *Movie A* (*like*), so y_{uv}=1. We first represent the interaction between users and items as triples (user, interact, item), and then connect the interaction data with the attribute information of items to form a complete knowledge graph G.

The problem to be solved in this paper is to obtain higher-order connection information from the higher-hop neighbors of the knowledge graph G. Then predict the probability $\hat{y}_{uv}$ of interaction between user u and item v (u and v have not interacted before).

3.2 Pretreatment Layer

The knowledge graph is usually preprocessed by knowledge graph embedding. This method can effectively parameterize entities and relationships into vectors while maintaining the structure of the knowledge graph. In triples, head entities and tail entities usually represent different types of entities. Such as the triple (*Titanic, DirectOf, James Cameron*) indicates the movie Titanic is directed by James Cameron. In this triple, the head entity *Titanic* is the movie name, and the tail entity *James Cameron* is a person. Therefore, entities should be mapped in a different way, and the mapping should be related to both entities and relationships. In this paper, we use the TransD [24] method to map entities and relationships. We define two vectors for each entity and relationship, one vector represents the meaning of entities or relationships, and the other vector (called the projection vector) is used to construct the mapping matrix. The network architecture of the pretreatment layer is shown in Fig. 3.

For triple (*h,r,t*), TransD learns each embedded entity and relationship by optimizing $\mathcal{E}_h + e_r \approx \mathcal{E}_t$, where $e_h, e_t \in \mathbb{R}^m$ represents the embedding of entities h and t, $e_r \in \mathbb{R}^n$ represents the embedding of relation r, m and n represent dimensions. For $\mathcal{E}_h = M_{rh}e_h$ and $\mathcal{E}_t = M_{rt}e_t$, $\mathcal{E}_h$ and $\mathcal{E}_t$ are the projected representations of e_h and e_t in relational space, M_{rh} and M_{rt} are mapping matrices, where $M_{rh}, M_{rt} \in \mathbb{R}^{m\times n}$, which are defined by entities and relationships. For M_{rh} and M_{rt}, $M_{rh} = r_p h_p^\top + I^{m\times n}$ and $M_{rt} = r_p t_p^\top + I^{m\times n}$, h_p and t_p represent the entity semantic vectors, r_p represents the relational semantic vector, where $h_p, t_p \in \mathbb{R}^m$, $r_p \in \mathbb{R}^n$, $I^{m\times n}$ is used to initialize each mapping matrix. For a given triple (*h,r,t*), its credibility score is defined as equation 2:

$$f(h,r,t) = \|\mathcal{E}_h + e_r - \mathcal{E}_t\|_2^2 \tag{2}$$

To better distinguish between valid triples and invalid triples during training, we use the loss of pairwise ranking to distinguish them. The formula is as shown in equation 3:

$$\mathcal{L}_{KG} = \sum_{(h,r,t,t')\in\mathcal{T}} -\ln\sigma\left(f\left(h,r,t'\right) - f\left(h,r,t\right)\right) \tag{3}$$

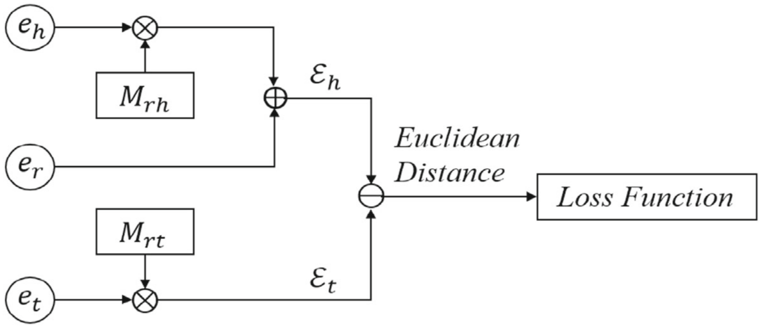

Fig. 3 Network architecture of the pretreatment layer

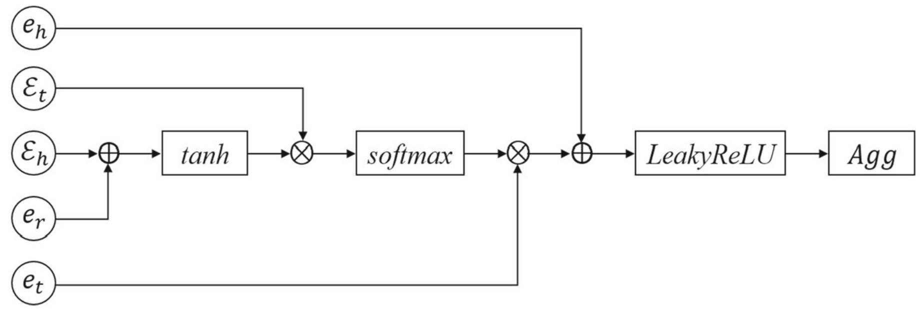

Fig. 4 Network architecture of the graph attention embedding layer

where $\mathcal{T} = \{(h, r, t, t')|(h, r, t) \in G, (h, r, t') \notin G\}$. (h, r, t') is an incomplete triple, which is constructed by randomly replacing a tail entity in the valid triple; $\sigma(\cdot)$ is a sigmoid function. This layer models the entities and relationships of triples and obtains their low-dimensional vector representations, thereby improving the representation capabilities of the model.

3.3 Graph Attention Embedding Layer

With the attention embedding layer, we can explicitly leverage first-order link information to associate users, items, and knowledge entity representations. The network architecture of the graph attention embedding layer is shown in Fig. 4.

For entity h, $h^t_{N_h}$ is used to represent its neighbor structure, where $N_h = \{(h, r, t)|(h, r, t) \in G\}$. We use the function $\delta(h, r, t)$ to represent the amount of information transmitted from t to h, and the number of this amount is determined by the relationship r. The formula is as shown in Eq. 4:

$$\delta(h, r, t) = (\mathcal{E}_t)^\top \tan h((\mathcal{E}_h + e_r)) \tag{4}$$

The nonlinear activation function in the formula uses *tanh*. The attention score depends on the distance between $\mathcal{E}_h$ and $\mathcal{E}_t$ in the relational r space and is able to transmit more information to closer entities. For the topological neighborhood structure of entity h, calculate the h's linear combination of neighborhoods at Eq. 5:

$$h^t_{N_h} = \Sigma_{(h,r,t)\in N_h} \tilde{\delta}(h, r, t)\, e_t \tag{5}$$

Normalize it with the *softmax* function, as shown in Eq. 6:

$$\tilde{\delta}(h, r, t) = \frac{\exp(\delta(h, r, t))}{\Sigma_{(h,r',t')\in\mathcal{N}_h} \exp(\delta(h, r', t'))} \tag{6}$$

Therefore, the final attention score can give a set of neighbor nodes that should pay more attention, and then capture more valuable cooperation signals. When forward propagation is performed, the data part of attention propagation can explain the recommendation results.

Finally, the entity representation e_h and its neighborhood representation $h^t_{N_h}$ are aggregated. In this paper, we use three types of aggregators:

(1) GCN aggregated [25]: Adds the two representations and performs a nonlinear transformation. The aggregation formula is as equation 7:

$$agg_{GCN} = LeakyReLU\left(W \cdot \left(e_h + h^t_{N_h}\right) + b\right) \tag{7}$$

The activation function is *LeakyReLU* [26], where $W \in \mathbb{R}^{m' \times m}$ is a trainable weight matrix to extract serviceable information for dissemination, and m' is the size of the transformation.

(2) GraphSage aggregated [19]: Connect the two representations first and then perform non-linear transformations. The aggregation formula is as equation 8:

$$agg_{GraphSage} = LeakyReLU\left(W \cdot \left(e_h \| h^t_{N_h}\right) + b\right) \tag{8}$$

where $\|$ represents the connection operation.

(3) GAT aggregated [27]: The neighborhood representation of entity h is represented directly as output. The aggregation formula is as equation 9:

$$agg_{GAT} = LeakyReLU\left(W \cdot h^t_{N_h} + b\right) \tag{9}$$

In AS-KGAN, an item is represented by aggregating its neighbors, so aggregation operation is important. The advantage of this layer is to explicitly use first-order connection information to associate users, items, and knowledge entity representations. In the experiment, we will evaluate these three aggregators.

Higher order propagation: The final representation of the entity consists of itself and its neighbors. We can further stack more propagation layers to obtain higher-order connection information. The AS-KGAN is extended from one layer to multiple layers so that the information propagated by higher-hop neighbors is collected. This method explores the potential interests of users in a broader, deeper, and more reasonable way. In step l, we recursively represent the entity as shown in equation 10:

$$e^{(l)}_u = \mathcal{Z}\left(e^{(l-1)}_h, h^{t(l-1)}_{N_h}\right) \tag{10}$$

The l-order neighborhood of entity h is defined as equation 11:

$$h^{t(l-1)}_{N_h} = \Sigma_{(h,r,t)\in N_h} \tilde{\delta}(h, r, t)\, e^{(l-1)}_t \tag{11}$$

where $e^{(l-1)}_t$ represents the entity t generated from the previous information propagation step and stores information from $(l-1)$ neighbors. $e^{(0)}_h$ is set as e_h of the initial information propagation iteration, which further promotes the representation of entity h at layer l.

3.4 Prediction Layer

After executing layer l, we get multiple representations of the entity h, and the formula represents $\{e^{(1)}_h, ..., e^{(l)}_h\}$. In this paper, user u represents entity h and item v represents entity t. Therefore, the multiple representation of user u is $\{e^{(1)}_u, ..., e^{(l)}_u\}$ and the multiple representation of item v is $\{e^{(1)}_v, ..., e^{(l)}_v\}$. The output of layer l is message aggregation with u (or v) as the root and l as tree structure depth. The outputs of different layers emphasize connectivity information of different sequences. For this reason, we use the layer aggregation mechanism proposed in document [28] to connect the representation of each step into a single vector, as shown in equation 12:

$$e^L_u = e^{(0)}_u \| \cdots \| e^{(l)}_u, e^L_v = e^{(0)}_v \| \cdots \| e^{(l)}_v \tag{12}$$

The $\|$ in the above formula is the splicing operation. The embedding propagation operations can enrich the initial embedding, and the propagation intensity can also be changed by adjusting l.

The goal of the model is to predict the probability that the user will interact with the item. We perform inner product operations on user and item representations to predict their match score in equation 13:

$$\hat{y}(u, v) = e_u^L \top e_v^L \tag{13}$$

The overall description of AS-KGAN is as follows:

Algorithm Personalized recommendation based on attribute significance of the knowledge graph attention network, AS-KGAN

```
Require: entity embeddings $e_h$ and $e_t$, relation embedding $e_r$, entity semantic vectors $h_p$ and $t_p$, relational
  semantic vector $r_p$. $e_h, e_t, h_p, t_p \in \mathbb{R}^m$, $e_r, t_p \in \mathbb{R}^n$
Ensure: $\hat{y}(u, v)$
 1: As to triples $(h, r, t)$
 2:     $M_{rh} \leftarrow r_p h_p^\top + I^{m\times n}$
 3:     $M_{rt} \leftarrow r_p t_p^\top + I^{m\times n}$
 4:     $\mathcal{E}_h \leftarrow M_{rh} e_h$
 5:     $\mathcal{E}_t \leftarrow M_{rt} e_t$
 6: As to $h_{N_h}^t$
 7:     $\delta(h, r, t) \leftarrow (\mathcal{E}_t)^\top \tan h\,((\mathcal{E}_h + e_r))$
 8:     $h_{N_h}^t \leftarrow \Sigma_{(h,r,t)\in N_h} \tilde{\delta}(h, r, t)\, e_t$
 9:     $\mathcal{Z} \leftarrow agg(e_h, h_{N_h}^t)$
10: For range $l$ do
11:     $e_h^{(l)} \leftarrow \mathcal{Z}\left(e_h^{(l-1)}, h_{N_h}^{t(l-1)}\right) // h_{N_h}^{t(l-1)} \leftarrow \Sigma_{(h,r,t)\in N_h} \delta(h, r, t)\, e_t^{(l-1)}$
12: end
13:     $e_h^L \leftarrow e_h^{(0)} \| \cdots \| e_h^{(l)}$
14:     $e_u^L, e_v^L \leftarrow e_h^L$
15: $\hat{y}(u, v) \leftarrow e_u^L \top e_v^L$
```

AS-KGAN is divided into three parts: (1) Pretreatment layer. The entities are mapped by the method of knowledge graph embedding. The projection representation of the entity is obtained in the relationship space. (2) Graph attention embedding layer. Neighbor representations of nodes embed updates through recursive propagation and learning the weight of item attributes. (3) Prediction layer. This layer aggregates the user representation and item representation of all propagation layers and outputs the predicted matching score.

3.5 Model Optimization

We use Bayesian Personalized Ranking [29] (BPR) to optimize AS-KGAN. BPR assumptions can reflect the interaction of observable items and the user preferences should be assigned a higher predictive value than the unobservable items. The complete loss function is shown in Eq. 14:

$$\begin{aligned}\mathcal{L}_{AS-KGAN} = &\sum_{(h,r,t,t')\in\mathcal{T}} -\ln\sigma\left(f\left(h, r, t'\right) - f(h, r, t)\right)\\ &+ \sum_{(u,v,j)\in\mathcal{O}} -\ln\sigma\left(\hat{y}(u, v) - \hat{y}(u, j)\right) + \lambda\|\Theta\|_2^2\end{aligned} \tag{14}$$

$\mathcal{O} = \{(u, v, j)|(u, v) \in \mathcal{R}^+, (u, j) \in \mathcal{R}^-\}$ represents the training set; $\mathcal{R}^+$ represents the observed (positive) interaction between user u and item j; $\mathcal{R}^-$represents the sampling unob-

Table 2 Statistics for three datasets

	MovieLens-20M	Amazon-book	Last.FM
#users	138,159	70,679	1872
#items	16,954	24,915	3846
#interactions	13,501,622	847,733	42,346
#entities	102,569	88,572	9366
#relations	32	39	60
#KG triples	499,474	2,557,746	15,518

served (negative) interaction set; $\sigma(.)$ represents the sigmoid function; Θ represents the parameter set of the model. To prevent overfitting, the L2 regularization method of λ parameterization of Θ is used.

4 Experiments

In this section, to verify the effectiveness of our model AS-KGAN, we conduct experiments on three public datasets and compared with the benchmark model on common evaluation indicators. To further understand the performance of AS-KGAN, we conduct ablation experiments and analyze the results.

4.1 Experiment Settings

In this subsection, we describe the datasets, baselines and benchmark models, and experimental parameter settings used for the experiments.

4.1.1 Datasets

To evaluate the effectiveness of AS-KGAN, considering the field, size, and sparsity of the dataset, we select three public datasets: MovieLens-20M, Amazon-book, and Last.FM. The statistics of the three datasets are shown in Table 2.

(1) MovieLens-20M: This dataset is widely used to contain rating data of multiple users on multiple movies. The dataset also includes movie metadata information and user attribute information.
(2) Amazon-book: The Amazon-review dataset [30] records users' comments on Amazon website products and is a classic dataset for recommendation systems. The category of goods is divided into Books, Electronics, Movies and TV, CDs and Vinyl, and other sub-datasets. This paper uses the Amazon-book sub dataset.
(3) Last.FM: It is a dataset about music recommendations. For each user in the dataset, it includes a list of their most popular artists and the number of times they were played. In addition, it also includes user application tags that can be used to build content vectors.

The positive sample instance is each observed user-item interaction. The negative sampling strategy is used to match the items that the user has not interacted with. For ensuring the quality of datasets, the 10-core setting is used when processing the above three datasets, reserving users and items with at least 10 interactions. DBpedia ontology knowledge base [31] is used to construct knowledge graphs for each dataset. Mapping the users, items, and attribute values of each dataset to the corresponding entities. We consider the triples of entities that are directly aligned with the item, whether the entity represents a subject or an object. At the same time, unlike the existing knowledge perception datasets that only provide first-order entities, we also consider triples containing second-order neighbor entities.

4.1.2 Baselines

The proposed AS-KGAN is compared with the path-based recommendation methods PER [7] and MCRec [8], embedding-based recommendation methods CKE [13] and UGRec [16], propagation-based recommendation methods RippleNet [17] and CKAN [18], and graph neural network-based recommendation methods KGCN [22] and DEKGCN [23]. We adopt the method mentioned in KGCN [22] to treat CKE as a collaborative filtering and structured knowledge module.

4.1.3 Experimental Parameter Settings

The performance evaluation recommended by *top-K* is completed through two commonly used evaluation protocols Recall [32] and NDCG [33]. The Recall value indicates the number of items in the test set that emerge in the user recommendation list. NDCG evaluates sorting performance by the item's position in the list. For each dataset, the training set is obtained by randomly selecting 80% of the interaction history of each user, and 10% of the interaction in the training set is randomly selected as the verification dataset to adjust the hyperparameters. The remainder data is the test set. In the *top-K* recommendation, the trained model is used to select the top k items with the highest prediction scores for each user in the test set.

For each user in the test set, we treat items that the user does not interact with as negative items. Each method outputs the user's preference score for all items except the positive items in the training set. The dimensions of the model are set to m, $n = 64$. We use the Adam optimizer to optimize the model, and the batch size is set to 1024. The grid search method is used to adjust the parameters, and the learning rate is set to 0.0001. We run all the experiments on a Linux machine with a Tesla V100 GPU. We use Python 3.6.7 and PyTorch to implement the AS-KGAN model.

4.2 Experimental Results and Analysis

We introduce the comparison of different recommended models and AS-KGAN in this section. We take the highest result with $K = 20$ as the best result for each model. The performance comparison results are presented in Table 3.

We have the following observations:

(1) Embedding-based recommendation models (CKE and UGRec) are better than path-based recommendation models (PER and MCRec). The reason is that path-based methods rely on predefined meta-paths. The quantity and quality of manually designed meta-paths are limited, which leads to relatively poor recommendation performance. This represents the effectiveness of embedding-based recommendations.

(2) Propagation-based recommendation models (RippleNet and CKAN) outperform embedding-based recommendation models. This proves that it is effective to explore users' potential preferences along the relationships in the knowledge graph. It also verifies the importance of merging adjacent items to enrich user representations and indicates that capturing adjacent information in knowledge graphs is effective for the recommendation.

Table 3 Overall performance comparison for $K = 20$

	MovieLens-20M		Amazon-book		Last.FM	
	Recall	NDCG	Recall	NDCG	Recall	NDCG
PER	0.2104	0.1608	0.1292	0.0853	0.1875	0.1598
MCRec	0.2137	0.1669	0.1330	0.0881	0.1921	0.1649
CKE	0.2188	0.1715	0.1383	0.0911	0.1925	0.1659
UGRec	0.2176	0.1694	0.1376	0.0907	0.1922	0.1653
RippleNet	0.2206	0.1742	0.1437	0.0955	0.1966	0.1703
CKAN	0.2279	0.1763	0.1441	0.0982	0.1998	0.1710
KGCN	0.2313	0.1778	0.1483	0.1021	0.2007	0.1716
DEKGCN	0.2322	0.1792	0.1487	0.1030	0.2015	0.1723
AS-KGAN	**0.2347**	**0.1809**	**0.1506**	**0.1042**	**0.2023**	**0.1739**

(3) The graph neural network-based recommendation models (KGCN, DEKGCN) have the best effect in all benchmarks, which shows that the application of the graph neural network makes the recommendation results more accurate. DEKGCN's prediction performance is slightly better than KGCN. The possible reason is that DEKGCN integrates the auxiliary information of users and items on the user side and the item side respectively. The utilization of heterogeneous information is helpful to improve the performance of the recommendation system.

(4) AS-KGAN produces the best performance on all datasets. The reason is that our model adopts the end-to-end approach in modeling, which can better obtain the high-order relationship in the knowledge graph. AS-KGAN is better than KGCN and DEKGCN because it considers the importance of attributes and constructs preferences for users' implicit feedback instead of using fixed weights. The datasets Amazon-book and Last.FM are sparser than the dataset MovieLens-20M. The improved performance of the model on datasets Last.FM and Amazon-book shows that it can handle sparse scenarios well.

We studied the influence of different K values on recommendation performance, where K = (1, 5, 10, 20, 50, 100). The experimental results in Fig. 5 show that the recommendation performance of our model is better than other models at different values of K.

4.3 Discussion

To further understand AS-KGAN, we conducted ablation experiments to discuss the embedding propagation layer numbers and aggregators on models.

4.3.1 Effect of Embedding Propagation Layer Numbers

We vary the depth of AS-KGAN to research the effect of embedding propagation layer numbers, and the number of layers is selected in the range of {1, 2, 3, 4}, the K value is set to 20, using the *GCN aggregator*. The experimental results are shown in Table 4.

As can be seen from the table, with the increase of l, Recall and NDCG values in the three datasets also increase. This is because the entity is embedded with semantic information about neighbor entities, which enriches the semantic information of entities. The higher-order relationships between users, items, and entities are effectively modeled. It can be observed

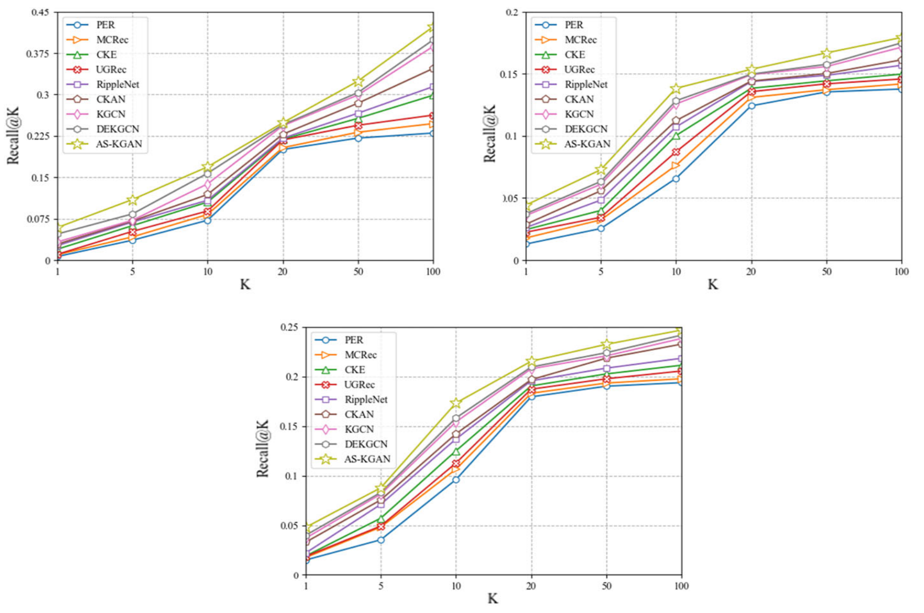

Fig. 5 The results of Recall@K in *top-K* recommendation

Table 4 Effect of embedding propagation layer numbers

	MovieLens-20M		Amazon-book		Last.FM	
	Recall	NDCG	Recall	NDCG	Recall	NDCG
1	0.2220	0.1712	0.1423	0.0981	0.1961	0.1680
2	0.2326	0.1793	0.1492	0.1033	0.2018	0.1734
3	**0.2347**	**0.1809**	**0.1506**	**0.1042**	**0.2023**	**0.1739**
4	0.2338	0.1802	0.1498	0.1036	0.2013	0.1730

that the model achieves its best performance when $l = 3$ and begins to decline when $l = 4$. This may be because the nearest neighbor entity far away from the entity contains less relevant information, which introduces noise and affects the performance of the recommendation. Therefore, this paper selects $l = 3$ as the length of multiple relational paths.

4.3.2 Effect of Aggregators

To research the influence of the aggregator on the model effect, the model depth is set to 1. Table 5 summarizes the experimental results.

As can be seen from the table, the GCN aggregator outperforms the GraphSage aggregator and GAT aggregator in all three datasets. This is probably because the GraphSage aggregator abandons the interaction between entity representation and entity ego-network. It means that the feature interaction is important when information aggregation and dissemination. The GAT aggregator only uses the neighbor representation and loses the useful information of the entity.

Table 5 Effect of aggregators

	MovieLens-20M		Amazon-book		Last.FM	
	Recall	NDCG	Recall	NDCG	Recall	NDCG
GCN	**0.2220**	**0.1712**	**0.1423**	**0.0981**	**0.1961**	**0.1680**
GraphSage	0.2217	0.1708	0.1420	0.0979	0.1957	0.1677
GAT	0.2216	0.1708	0.1417	0.0977	0.1954	0.1675

5 Conclusions and Future Work

To research the problem of item attribute significance on the knowledge graph, we proposed a personalized recommendation model based on the attribute significance of the knowledge graph attention network. In the knowledge graph attention network with attribute significance, the knowledge graph is introduced into recommendation as auxiliary information, which effectively alleviates the problems of data sparsity and cold start. When constructing the knowledge graph, we not only consider user attribute information and item attribute information but also consider the influence of the importance of the item's different attributes on the recommendation. Experimental results on MovieLens-20M, Amazon-book, Last.FM public datasets prove the validity of our proposed model. In the future, we will further combine the knowledge graph with other auxiliary information (Such as user comment information and user trust relationship, etc.) effectively and apply it in the recommendation system to improve the accuracy of personalized recommendations. We can also combine social networks to research the influence of social relationships on recommendation results to further improve the interpretability of recommendations.

Acknowledgements The authors are grateful to the editors and reviewers for their helpful comments and suggestions. This research is partially supported by National Social Science Foundation project (17BXW065), National Key R&D Program of China (2018******01).

References

1. Cheng L, Gao MT (2019) Hybrid recommendation algorithm based on time weighted and LDA clustering [J]. Computer Eng Appl 55(11):160–166
2. Jamali M, Ester M (2010) A matrix factorization technique with trust propagation for recommendation in social networks. Proceedings of the Fourth ACM Conference on Recommender Systems, 135–142. https://doi.org/10.1145/1864708.1864736
3. Wang HW, Zhang FZ, Hou M et al (2018) Shine: Signed heterogeneous information network embedding for sentiment link prediction. Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, 592–600. https://doi.org/10.1145/3159652.3159666
4. Bansal T, Belanger D, Mccallum A (2016) Ask the GRU: Multi-Task Learning for Deep Text Recommendations. Proceedings of the 10th ACM Conference on Recommender Systems, 107-114. https://doi.org/10.1145/2959100.2959180
5. Sharma A, Cosley D (2013) Do social explanations work? Studying and modeling the effects of social explanations in recommender systems. Proceedings of the 22nd international conference on World Wide Web, 1133-1144. https://doi.org/10.1145/2488388.2488487
6. Chang L, Zhang WT, Gu TL et al (2019) Review of recommendation systems based on knowledge graph. CAAI Trans Intell Syst 14(2):207–216. https://doi.org/10.11992/tis.201805001
7. Yu X, Ren X, Sun YZ, et al. (2014) Personalized entity recommendation: A heterogeneous information network approach. Proceedings of the 7th ACM International Conference on Web Search and Data Mining, 283-292. https://doi.org/10.1145/2556195.2556259

8. Hu BB, Shi C, Zhao WX, et al. (2018) Leveraging meta-path based context for top-n recommendation with a neural co-attention model. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery And Data Mining, 1531-1540. https://doi.org/10.1145/3219819.3219965
9. Zhao H, Yao QM, Li JD, et al. (2017) Meta-graph based recommendation fusion over heterogeneous information networks. Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 635-644. https://doi.org/10.1145/3097983.3098063
10. Wang X, Wang DX, Xu CR et al (2019) Explainable reasoning over knowledge graphs for recommendation. Proceedings of the AAAI Conference on Artificial Intelligence. 33(01):5329–5336. https://doi.org/10.1609/aaai.v33i01.33015329
11. Wang X, Huang TL, Wang DX et al (2021) Learning Intents behind Interactions with Knowledge Graph for Recommendation. Proceedings of the Web Conference. 878–887. https://doi.org/10.1145/3442381.3450133
12. Wang Q, Mao Z, Wang B et al (2017) Knowledge graph embedding: a survey of approaches and applications. IEEE Trans Knowl Data Eng 29(12):2724–2743. https://doi.org/10.1109/TKDE.2017.2754499
13. Zhang FZ, Yuan NJ, Lian D, et al (2016) Collaborative knowledge base embedding for recommender systems. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 353-362. https://doi.org/10.1145/2939672.2939673
14. Wang HW, Zhang FZ, Xie X, et al (2018) DKN: Deep knowledge-aware network for news recommendation. Proceedings of the 2018 World Wide Web Conference, 1835-1844. https://doi.org/10.1145/3178876.3186175
15. Nathani D, Chauhan J, Sharma C, et al (2019) Learning attention-based embeddings for relation prediction in knowledge graphs. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1466
16. Zhao XX, Cheng ZY, Zhu L, et al. (2021) UGRec: Modeling directed and undirected relations for recommendation. Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, 193-202. https://doi.org/10.1145/3404835.3462835
17. Wang HW, Zhang FZ, Wang JL, et al. (2018) RippleNet: Propagating user preferences on the knowledge graph for recommender systems. Proceedings of the 27th ACM International Conference on Information and Knowledge Management, 417-426. https://doi.org/10.1145/3269206.3271739
18. Wang Z, Lin GY, Tan HB, et al. (2020) CKAN: Collaborative knowledge-aware attentive network for recommender systems. Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 219-228. https://doi.org/10.1145/3397271.3401141
19. Hamilton WL, Ying R, Leskovec J (2017) Inductive representation learning on large graphs. Proceedings of the 31st International Conference on Neural Information Processing Systems, 1025-1035
20. Wang J (2019) Graph neural network analysis. Modern Computer 23:58–62
21. Berg R, Kipf TN, Welling M (2018) Graph convolutional matrix completion. Proceedings of KDD'18 Deep Learning Day, August 2018, London, UK. https://doi.org/10.48550/arXiv.1706.02263
22. Wang HW, Zhao M, Xie X et al (2019) Knowledge graph convolutional networks for recommender systems. Proceedings of World Wide Web Conference, New York: ACM Press, 3307–3313. https://doi.org/10.1145/3308558.3313417
23. Li X, Yang XY, Yu J et al (2022) Double end knowledge graph convolutional networks for recommender systems. J Front Computer Sci Technol 16(1):176. https://doi.org/10.3778/j.issn.1673-9418.2103072
24. Ji GL, He SZ, Xu LH, et al. (2015) Knowledge graph embedding via dynamic mapping matrix. 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), 687-696
25. Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. International Conference on Learning Representations (ICLR), 2017. https://doi.org/10.48550/arXiv.1609.02907
26. Maas AL, Hannun AY, Ng AY (2013) Rectifier nonlinearities improve neural network acoustic models. In ICML, Vol. 30. 3
27. Veličković P, Cucurull G, Casanova A, et al. (2018) Graph attention networks. In Proceedings of the 6th International Conferences on Learning Representations, 2018. https://doi.org/10.48550/arXiv.1710.10903
28. Xu K, Li C, Tian Y, et al. (2018) Representation learning on graphs with jumping knowledge networks. International Conference on Machine Learning. PMLR, 5453-5462
29. Rendle S, Freudenthaler C, Gantner Z, et al. (2009) BPR: Bayesian personalized ranking from implicit feedback. In UAI. 452-461. https://doi.org/10.48550/arXiv.1205.2618
30. He R, McAuley J (2016) Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. Proceedings of the 25th International Conference on World Wide Web, 507-517. https://doi.org/10.1145/2872427.2883037

31. Bizer C, Lehmann J, Kobilarov G et al (2009) DBpedia: a crystallization point for the web of data. Web Seman Sci Serv Agents on the World Wide Web 7(3):154–165. https://doi.org/10.1016/j.websem.2009.07.002
32. He XN, Liao LZ, Zhang HW, et al. (2017) Neural collaborative filtering. Proceedings of the 26th International Conference on World Wide Web, 173-182. https://doi.org/10.1145/3038912.3052569
33. Yang JH, Chen CM, Wang CJ, et al. (2018) HOP-rec: high-order proximity for implicit recommendation. Proceedings of the 12th ACM Conference on Recommender Systems, 140-144. https://doi.org/10.1145/3240323.3240381