Applied Intelligence (2022) 52:9308–9319
https://doi.org/10.1007/s10489-021-02758-9

EKPN: enhanced knowledge-aware path network for recommendation

Peng Yang[1,2,3] · **Chengming Ai**[1,2,3] · **Yu Yao**[1,2,3] · **Bing Li**[1,2,3]

Accepted: 7 August 2021 / Published online: 4 January 2022

Abstract

Incorporating a knowledge graph (KG) into recommender systems has been widely studied by researchers. Existing methods mostly extract the paths connecting user-item pairs and model these paths or iteratively propagate the user preference over the KG. These methods can capture the semantics of entities and relations well and help to comprehend users' interests. However, these methods ignore the implicit features between the KG's external items and thus cannot fully capture the users' preferences. To solve this problem, we propose a novel model named **E**nhanced **K**nowledge-aware **P**ath **N**etwork (EKPN) to exploit the KG and capture implicit features between items outside the KG for recommendation. EKPN consists of two neural network modules: one module captures explicit features between items in the knowledge graph by automatically generating paths from users to candidate items for recommendation; the other module explores implicit features between items outside the knowledge graph by utilizing users' historical interactions. To better capture the implicit features between items outside the knowledge graph, we propose the activation gate mechanism. Finally, we use a fusion mechanism to combine the two modules to enhance each other and achieve higher performance. Extensive validation on two real-world datasets shows the superiority of EKPN over baselines.

Keywords Recommendation · Knowledge graph · Implicit features · Collaborative filtering

1 Introduction

Recommendation systems (RS) play an important role in various web applications, such as e-commerce, news portals, social media sites, and so on. Recommendation systems use a variety of recommendation technologies. The collaborative filtering (CF) algorithm has been widely used in recommendation systems, which uses users' historical interactions and makes recommendations based on the user's common historical interaction. Although the collaborative filtering method is effective and universal, it cannot model side information, such as item attributes, user profiles, and contexts. It has been found that collaborative filtering methods are mainly based on the interaction between the user and the item. If users and items rarely interact with sparse situations, the performance of collaborative filtering method will be poor. To solve this problem, the common paradigm is to transform the side information into a vector and feed it into the model to predict the score. These methods have been widely used in industry, such as factorization machines (FMs) [1], wide & deep [2], DeepFM [3] and xDeepFM [4]. The FM method models pairwise feature interactions as inner products of latent vectors between the features and shows very promising results. While in principle, FM can model high-order feature interaction, in practice, usually only order-2 feature interactions are considered due to high complexity. Therefore, wide & deep [2], DeepFM [3] and xDeepFM [4] use a DNN to capture high-order feature interactions and FM to capture low-order feature interactions, but these methods have weak interpretability. The knowledge graph contains rich side information and is a type of directed heterogeneous graph in which the nodes correspond to entities and the edges correspond to relations. Therefore, the method can be used to discover commonalities between

✉ Peng Yang
pengyang@seu.edu.cn

1 School of Computer Science and Engineering, Southeast University, Nanjing, China

2 School of Cyber Science and Engineering, Southeast University, Nanjing, China

3 Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education, Nanjing, China

items, and we can also use a knowledge graph to solve this problem.

There are many recommendation methods based on knowledge graphs, and these methods can be roughly categorized into two types: path-based methods and embedding-based methods. The embedding-based method leverages knowledge graph embedding (KGE) techniques to learn the representation of items and users. For instance, CKE [5] adopts TransR [6] to capture semantic information in KG and extracts the items' structural representations. However, KGE techniques are more suitable for link prediction and inference tasks rather than recommendation. Therefore, RippleNet [7] expands the users' interests in knowledge graphs and discovers users' potential interests, but the significance of connectivity is weakly characterized in the framework. KGCN [8] and KGCN-LS [9] use the node's neighbours to compute the user's and item's vectors in KG by using graph convolutional neural (GCN) [10] networks. However, explicit collaborative signals are ignored in both methods, which results in insufficient item embeddings. Recently, KGAT [11] used a graph attention network (GAT) [12] to compute user and item vectors through a KG. Overall, these methods use representation learning to establish a connection between users and items in an implicit manner, rather than inferring user preferences. Although these methods can improve the recommendation performance, they capture only the direct relationship between the entities and do not consider the multilayer relationships between the entities.

Path-based methods mainly refine the similarity of the meta-path connected between the users and the items, such as SemRec [13]. However, meta-paths are inefficient in KG and have two issues: 1) meta-paths cannot capture semantic information since they do not contain relational information; 2) meta-paths cannot automatically uncover and reason for unseen connectivity patterns because they need to predefine domain knowledge. To solve these problems, RKGE [14] and KPRN [15] use RNNs to model these paths by automatically extracting the paths between user-items. However, these methods lose much information in the automatic extraction of the paths between user-items and can only capture the explicit features, which are the relationships between items in the knowledge graph. There are many explicit features between the movies in the movie domain, including movies directed by the same director, movies played by the same actor, etc. The relationship between the movie may be much broader than the knowledge graph, and the knowledge graph cannot contain all the relationships between items, so the current knowledge-aware recommendation method has difficulty capturing the implicit features, which are the relationships between items outside the knowledge graph, including two movies that obtained an award at the same time, two movies that were created by the same screenwriter or two movies that are of the same type.

Prior efforts on knowledge-aware recommendation extracts the explicit features between items contained in the knowledge graph and cannot capture the implicit features between items outside the knowledge graph. As shown in Fig. 1, although there are no explicit features between some items of the user's historical and candidate items, there may be implicit features, which can assist the model in discovering the potential interests of users outside the knowledge graph and enhancing the recommended performance. To

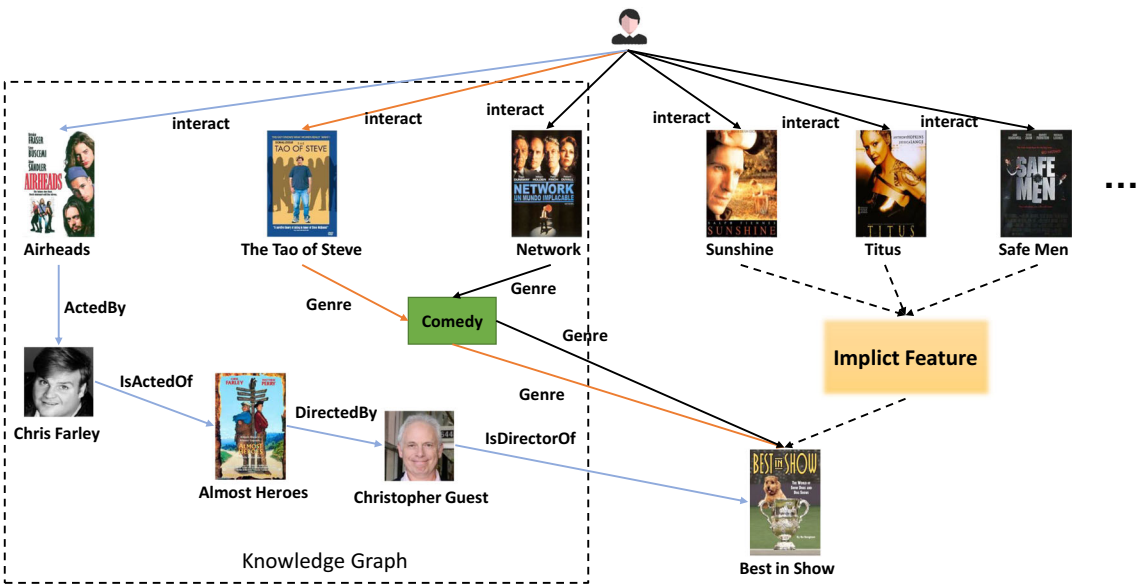

Fig. 1 Illustration of KG-aware recommendation in the movie domain

this end, we propose a novel model named the enhanced knowledge-aware path network (EKPN) to solve the problem that prior efforts cannot capture the implicit features between items outside the knowledge graph. EKPN contains two modules: one module captures the explicit features between items in the knowledge graph for recommendation by generating the path from the user to the candidate item automatically; the other module explores implicit features between items outside the knowledge graph by utilizing the users' historical interactions. To enhance both of these modules, a fusion mechanism is used to combine the two modules for recommendation. The contributions of this work are three-fold:

1. We propose an end-to-end model that can capture implicit features between items outside the knowledge graph and learn path se mantics in the knowledge graph for recommendation.
2. To better capture the implicit features between items outside the knowledge graph, we propose an activation gate mechanism that can automatically learn the correlation score between items, thereby capturing implicit features between the items.
3. We conduct extensive experiments, and the results on two publicly available datasets demonstrate the effectiveness of our proposed model. Moreover, we analyse the effect of different model components on the overall performance and use a running example to illustrate the interpretability of our model.

The structure of the paper is as follows. In Section 2, we introduce some relevant and recently published work related to the topic of the paper; in Section 3, we introduce some definitions that are used throughout the paper; in Section 4, we provide a detailed description of the experimental setup. We also report and discuss the experimental results, and in Section 5, we summarize the findings of the paper.

2 Related work

With the explosive growth of information, personalized recommendations for satisfying user preferences have become increasingly important. Many previous works try to solve the problem of personalized recommendation, but these methods have weaknesses in interpretability, such as CF-based methods [16], content-based methods [17] and hybrid-based [18] methods.

In recent years, many researchers have tried to use knowledge graphs to solve the problem of interpretability of recommendations. This method can be divided into two types: embedding-based methods and path-based methods. The embedding-based method learns item and user representations with KGE [6, 19] by using the relationships between entities directly connected in the KG. For example, CKE [5] adopts TransR [6] to capture semantic information in the KG and extracts items' structural representations. However, KGE techniques are more suitable for link prediction and inference tasks rather than recommendation. Therefore, RippleNet [7] discovers users' potential preference by extending the relationships in the KG automatically and iteratively, but the importance of relations is weakly characterized in RippleNet. As GCN [10] appeared, some works [8, 9] used GCN to compute the node's vector in a KG using the node's neighbours. However, both methods ignore explicit collaboration signals and cannot fully learn the embedding of items. Recently, KGAT [11] has used GAT [12] to model the high-order connectivities in a KG. However, this regularization of knowledge graph embedding has not fully exploited the connectivity characterization between users and items. Their indirect utilization of the knowledge graph structure prevents them from effectively modelling the connectivity patterns and sequential dependencies. As a result, their recommendation accuracy and reasoning capability are limited. For example, they cannot explain their recommendation by providing a path in the KG that connects a user with the recommended item.

To solve the above problems, the path-based method introduces path connectivity information in a KG for recommendation learning. Early path-based methods [13] calculate the meta-path similarities between the user and the item to assist recommendation. This type of method requires predefined domain knowledge and heavily relies on the quality of the meta-path. In recent years, many studies have tried to infer user preferences by studying the path representations between users and items. For example, RKGE [14] learns the representation of the paths between entities to discover user preferences automatically and captures multiple layers of semantic information. However, this method only involves entity embeddings in path modelling, and these limitations may hurt the reasoning ability of the model. Therefore, KPRN [15] learns the user's interest preferences by extracting paths and using RNNs to model the paths automatically and consider the sequential dependencies, as well as relation semantics, to infer the user-item interaction. However, KPRN can capture only the explicit features contained in KG, and it is difficult to discover the implicit features outside the knowledge graph, causing the method to lose a large amount of useful information and resulting in poor recommendation. EKPN can discover implicit features between items to assist recommendation by utilizing the users' historical interactions.

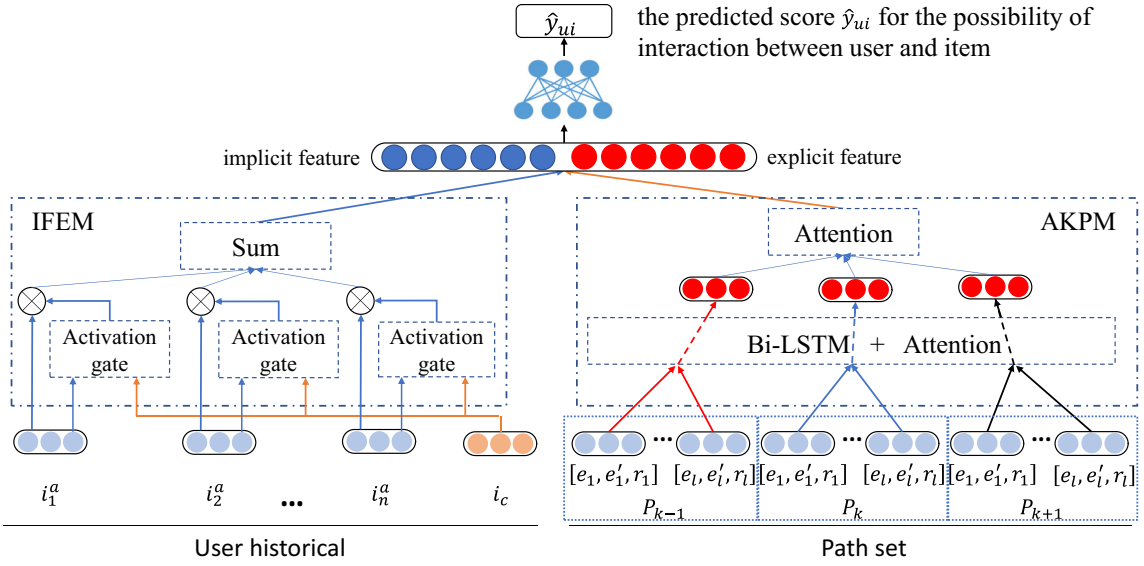

Fig. 2 Illustration of our EKPN model for recommendation. It consists of an implicit feature extraction module (IFEM) and an attention knowledge-aware path module (AKPM)

3 Approach

In this section, we first define the KG and user-item data and formulate the recommendation problem, then explain how to combine them as the inputs of our model and next introduce our proposed method in detail, as shown in Fig. 2.

3.1 Background

Table 1 summarizes all important notations used in this paper. We define the set of entities as $E = \{e_1, e_2, ..., e_k\}$ and relations between entities $R = \{r_1, r_2, ..., r_g\}$. Let an undirected graph $KG = \{(h, r, t)|h, t \in E, r \in R\}$ denote the domain KG, where (h, r, t) is a three triplet and r indicates a relationship from head entity h to tail entity t. Additionally, we define $U = \{u_1, u_2, ..., u_m\}$ as the set of users and $I = \{i_1, i_2, ..., i_n\}$ as the set of items, and use $\mathcal{V} \in R^{m \times n}$ to denote historical user-item interactions, where $v_{ij} = 1$ indicates that there is an interaction between the user u_i and the item i_j, $v_{ij} = 0$ otherwise. Following [20], if there is a historical interaction between a user and an item, we represent the interaction with a triplet $\tau = (u, interact, i)$, where $interact$ is a predefined relation. To use the user-item interaction data and domain KG together, we use the string match method [15] to merge items and entities. Therefore, $G = \{(h, r, t)|h, t \in E', r \in R'\}$ denotes the KG that merges the user-item interaction data and domain KG.

Table 1 Notations

Notations	Descriptions
$E = \{e_1, e_2, ..., e_k\}$	Entity set
$R = \{r_1, r_2, ..., r_g\}$	Relation set between entities
$KG = \{(h, r, t)\|h, t \in E, r \in R\}$	Undirected graph
$U = \{u_1, u_2, ..., u_m\}$	User set
$I = \{i_1, i_2, ..., i_n\}$	Item set
$P(u, i) = \{p_1, p_2, ..., p_t, ..., p_m\}$	Path set
$y_u = \{i_1^u, i_2^u, ..., i_n^u\}$	User's historical interaction
$p_k = \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_L}\}$	The k-th path
$\mathbf{e_l} \in \mathbf{R}^d$	Entity embedding vector
$\mathbf{e_l'} \in \mathbf{R}^d$	Entity type embedding vector
$\mathbf{r_l} \in \mathbf{R}^d$	Relation embedding vector
$\overrightarrow{\mathbf{h}_l}$	Forward hidden states
$\overleftarrow{\mathbf{h}_l}$	Forward hidden states
$\mathbf{a_{ui}^p}$	The vector output by AKPM
$\mathbf{a_{ui}^I}$	The vector output by IFEM
α_i^p	The i-th path score in attention pooling

3.2 Path mining

To fully exploit the relationship of the entities and capture multiple layers of semantic information in the KG, we use the method in [15] and define a path that consists of all entities and relationships between user u and item i. We define $P(u, i) = \{p_1, p_2, ..., p_t, ..., p_m\}$ as the set of paths between user u and item i, where p_t represents the t-th path and defines the format of p as follows: $p = [e_0 \xrightarrow{r_1} e_1 \xrightarrow{r_2} e_2, ..., e_{L-1} \xrightarrow{r_{L-1}} e_L]$, where e_0 is user u, e_L is item i and L is the length of path. To reduce data noise, we only use paths with lengths less than a threshold. Sun [21] points out that paths with relatively short lengths can better model entity relations, while longer paths may lead to the inclusion

of noise due to the introduction of remote entities with less semantic relevance.

3.3 Task definition

We define the task as given a user u and an item i, the user's historical interaction $y_u = \{i_1^u, i_2^u, ..., i_n^u\}$ and a set of paths $P(u, i) = \{p_1, p_2, ..., p_m\}$ to obtain the possible score of user and item interaction.

$$\hat{y}_{ui} = f_\theta(u, i|P(u, i), y_u) \quad (1)$$

where θ is the parameter of model f and $\hat{y}_{ui}$ represents the predicted score for the possibility of interaction between user and item.

3.4 Model

In this section, we present the EKPN model for the recommendation task; the model architecture is illustrated in Fig. 2 and consists of two main components: 1) an attention knowledge-aware path module (AKPM) and 2) an implicit feature extraction module (IFEM).

3.4.1 Attention knowledge-aware path module

As shown in Fig. 3, AKPM focuses on learning representations for paths in the KG. AKPM uses a set of paths $P(u, i) = \{p_1, p_2, ..., p_m\}$ from user u to item i as input. We use $p_k = \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_L}\}$ to represent the k-th path in

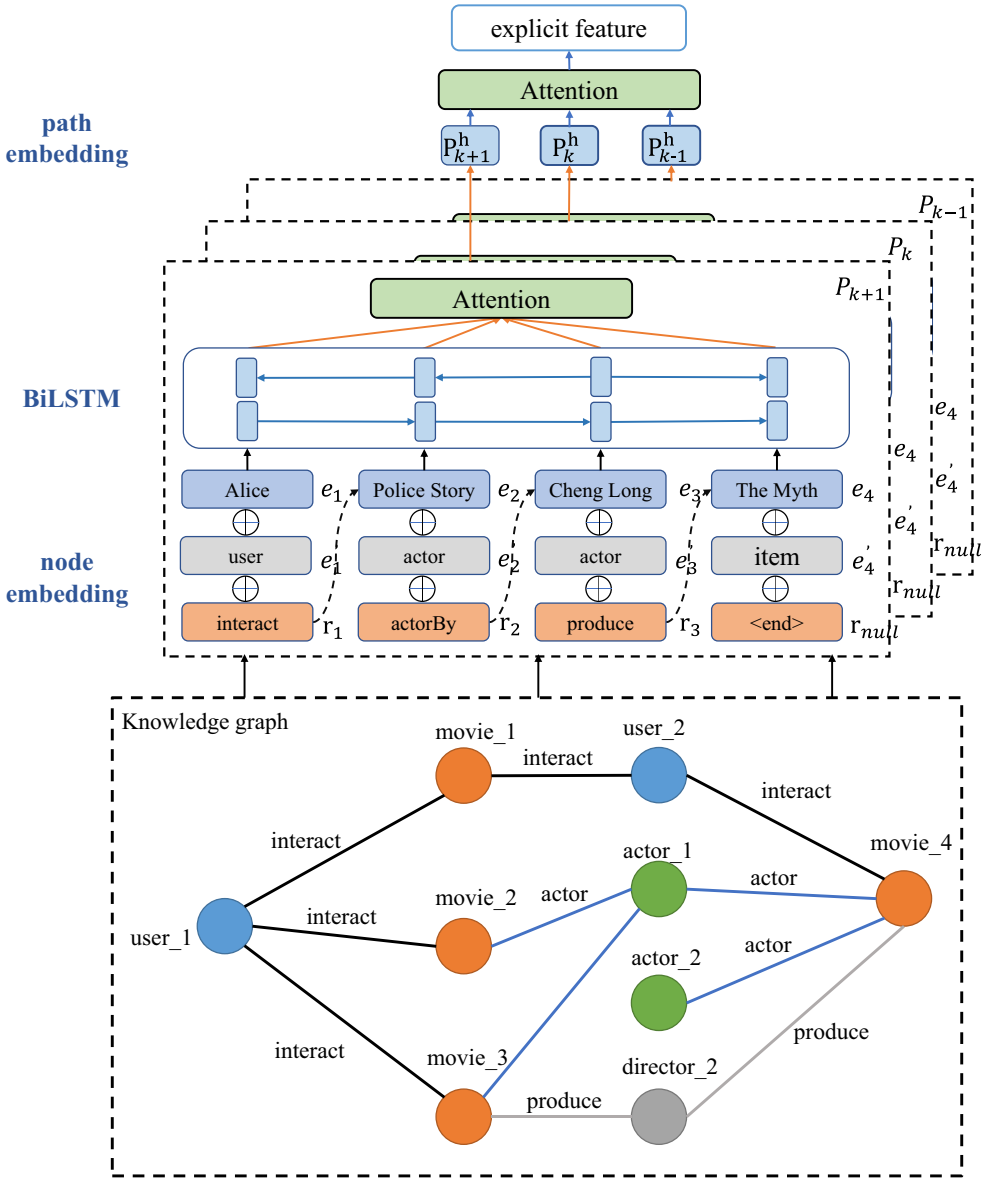

Fig. 3 Illustration of our AKPM component and the embedding layer consists of three separate layers for entity, entity type and relation type

the path set.

$$\mathbf{x_l} = \mathbf{e_l} \oplus \mathbf{e'_l} \oplus \mathbf{r_l} \tag{2}$$

where $\mathbf{e_l} \in \mathbf{R}^d$ and $\mathbf{e'_l} \in \mathbf{R}^d$ are two separate embedding vectors, $\mathbf{e_l}$ represents the entity in the path, $\mathbf{e'_l}$ represents the entity type and $\mathbf{r_l} \in \mathbf{R}^d$ represents the relationship between the entities; $\mathbf{d}$ is the embedding size and $\mathbf{l}$ is the l-th node in the path; $\oplus$ is the concatenation operation. For the last input $\mathbf{x_L}$ in the path, a null relation $\mathbf{r_{null}}$ is used as padding. To explore the sequential information for the path, we adopt bidirectional long short-term memory (Bi-LSTM)+attention as the base model.

$$\begin{aligned} \overrightarrow{\mathbf{h}_l} &= \overrightarrow{LSTM}(\mathbf{x}_l, \overrightarrow{\mathbf{h}_{l-1}}), \\ \overleftarrow{\mathbf{h}_l} &= \overleftarrow{LSTM}(\mathbf{x}_l, \overleftarrow{\mathbf{h}_{l+1}}). \end{aligned} \tag{3}$$

The Bi-LSTM generates the forward hidden states $\overrightarrow{\mathbf{h}_l}$ and backward hidden states $\overleftarrow{\mathbf{h}_l}$ at each time step l. Then, $\overrightarrow{\mathbf{h}_l}$ and $\overleftarrow{\mathbf{h}_l}$ are concatenated to obtain the hidden states $\mathbf{h}_l = [\overrightarrow{\mathbf{h}_l}, \overleftarrow{\mathbf{h}_l}]$ at time step l, where $\mathbf{h}_l \in \mathbf{R}^{d_h}$. The final output sequence of the Bi-LSTM is $\mathbf{H} = (\mathbf{h}_1, ..., \mathbf{h}_L)$. To aggregate the variable-size encoding sequence $\mathbf{H}$ to be a fixed-size path context vector $\mathbf{p_k^h}$, we use attention pooling (4, 5) to perform this task.

$$\boldsymbol{\alpha} = softmax(\frac{(\mathbf{q}\mathbf{W}_q)^T \mathbf{W}_k \mathbf{I}}{\sqrt{d_q}}) \tag{4}$$

$$\mathbf{O} = \boldsymbol{\alpha}\mathbf{W}_v\mathbf{I} \tag{5}$$

where $\mathbf{I}$ is an input matrix, $\mathbf{W}_q$, $\mathbf{W}_k$ and $\mathbf{W}_v$ are query, key and value parameter matrices, respectively, $\mathbf{q} \in \mathbf{R}^{d_q}$ is a trainable query vector, and $\mathbf{O}$ is the vector output by attention pooling. Therefore, we can use $\mathbf{H}$ as input and obtain a set of vectors for paths, which is given by $\mathbf{P^h} = (\mathbf{p_1^h}, ..., \mathbf{p_m^h})$. To explore the relation between paths, we use $\mathbf{P^h}$ as the input for attention pooling and obtain an output vector $\mathbf{a_{ui}^p}$. The final prediction is defined as:

$$\hat{y}_{ui} = sigmoid(\mathbf{W_{out}}\mathbf{a_{ui}^p}) \tag{6}$$

where $\mathbf{W_{out}}$ is a parameter matrix.

In summary, the attention knowledge-aware path module (AKPM) is used in this paper to capture the information in the KG.

3.4.2 Implicit feature extraction module

IFEM focuses on capturing implicit features between items outside the KG by utilizing the users' historical interactions, as shown in Fig. 4. Therefore, we use u_a's historical $y_a = \{i_1^a, i_2^a, ..., i_n^a\}$ and candidate item i_c as input, where i_j^a represents the ID of the j-th item interacted by user a. V = $\{\mathbf{v_1}, \mathbf{v_2}, ..., \mathbf{v_n}\}$ and $\mathbf{v_c}$ are used to transform the items into a low-dimensional vector that represents the vector for users' histories and the vector for the candidate items.

We design a novel activation gate mechanism that is able to compute a score between each item in the user's historical interaction and a candidate item to capture implicit features. The formula of the gate mechanism is defined as:

$$\begin{aligned} \mathbf{a_1} &= \mathbf{v_j} \oplus (\mathbf{v_j} \odot \mathbf{v_c}) \oplus \mathbf{v_c} \\ w_j &= sigmoid(\mathbf{W_{out}}(relu(\mathbf{W_2}\mathbf{a_1} + b_2)) + b_3) \end{aligned} \tag{7}$$

where $\odot$ represents the dot product, $\mathbf{W_2}$ and $\mathbf{W_{out}}$ are parameter matrices, $\mathbf{b_2}$ and $\mathbf{b_3}$ are parameter vectors, and w_j represents the score between item j and item c. Having obtained the score set $W = \{w_1, w_2, ..., w_n\}$, we aim to capture an implicit feature vector between the users' interaction historical and candidate items. Towards this end, sum pooling and two fully connected layers are adopted to

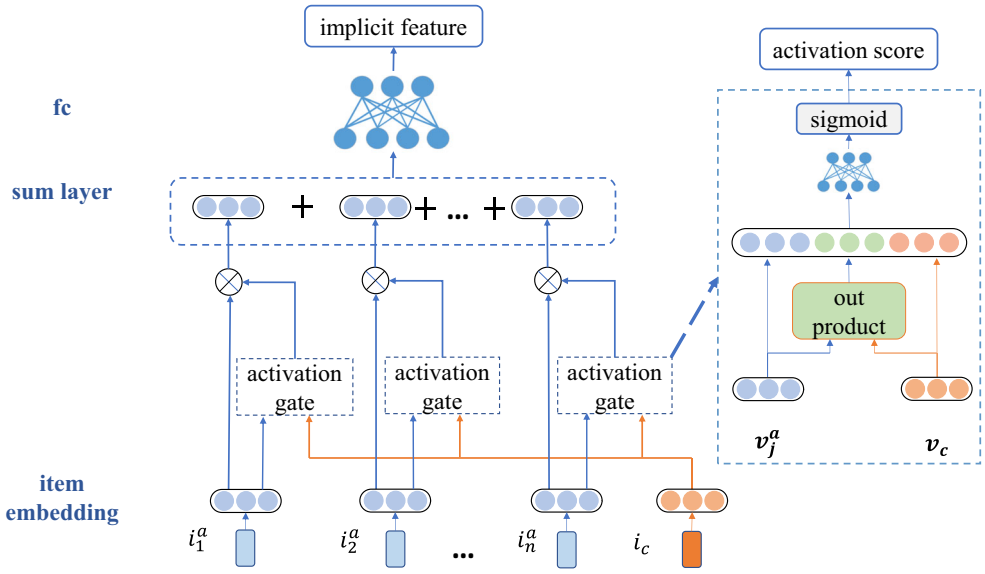

Fig. 4 Illustration of our IFEM component

capture the implicit feature.

$$\begin{aligned}\mathbf{v_u} &= \sum_{j=1}^{n} w_j \mathbf{v_j} \\ \mathbf{a_{ui}^I} &= relu(\mathbf{W_2}(\mathbf{v_u} \oplus \mathbf{v_c}) + b_2) \\ \hat{y}_{ui} &= sigmoid(\mathbf{W_{out}}\mathbf{a_{ui}^I} + b_3)\end{aligned} \tag{8}$$

where $\mathbf{W_2}$ and $\mathbf{W_{out}}$ are parameter matrices, $\mathbf{b_2}$ and $\mathbf{b_3}$ are parameter vectors, and $\mathbf{a_{ui}^I}$ is the implicit feature vector between the users' interaction historical and candidate items.

In summary, the implicit feature extraction module (IFEM) is used in this paper to capture the implicit feature vector between items.

3.4.3 Fusion

In the above, we introduced two components, AKPM and IFEM. To merge these two modules, we need to design a module to fuse them so that they can enhance each other. A common fusion strategy is to concatenate the learned features to obtain a joint representation and then feed it into a fully connected layer. In this paper, for AKPM, the feature vector between the user and candidate items in the knowledge graph is mainly extracted; for IFEM, the implicit vector between the user and candidate item is mainly extracted. These two vectors can be regarded as the representation of the corresponding user-item pairs. Then, the output of the fusion model can be defined as:

$$\hat{y}_{ui} = sigmoid(\mathbf{W_{out}}[\mathbf{a_{ui}^P} \oplus \mathbf{a_{ui}^I}]) \tag{9}$$

Since the two types of methods have different advantages and can learn the prediction vector from different perspectives, the concatenation of the two prediction vectors will provide a more powerful and robust joint representation.

3.5 Optimization

3.5.1 Loss function

Similar to previous work [7, 22], we convert the recommendation task into a binary classification task, where the target value 1 represents a user-item interaction that is observed and 0 represents otherwise. For the binary classification problem, negative log-likelihood is used as the objective function, which is defined as:

$$\mathcal{L} = -(\sum_{(u,i)\in\mathcal{O}^+} log(\hat{y}_{ui}) + \sum_{(u,j)\in\mathcal{O}^-} log(1-\hat{y}_{uj})) \tag{10}$$

where $O^+ = \{(u,i)|v_{ui} = 1\}$ is the positive user-item interaction pairs and $O^- = \{(u,j)|v_{uj} = 0\}$ is the negative user-item interaction pairs.

3.5.2 Pre-training

According to [23], the convergence and performance of a model not only is affected by the model itself but also depends on the initialization of the model parameters, and pre-training is a better method of model initialization, so the initialization of the integrated model using pre-trained weights can improve the convergence speed and model performance. Since EKPN consists of two components, AKPM and IFEM, we can pre-train these two components and use them to initialize EKPN.

Algorithm 1 describes the detailed optimization process. The algorithm is mainly composed of two modules: data preprocessing (lines 1-6) and an enhanced knowledge-aware path network (lines 8-24).

Algorithm 1 EKPN optimization.

Input: knowledge graph $\mathcal{G}$, rating matrix $\mathcal{V}$
1 **foreach** $u_j \in U$ **do**
2 Based on $\mathcal{V}$, get positive pairs (u_j, i_p) and user' historical $y_u = \{i_1^u, i_2^u, ..., i_n^u\}$;
3 Generate negative pairs (u_i, i_n) through random sampling;
4 $(u,i) \leftarrow (u_i, i_p) \cup (u_i, i_n)$;
5 **foreach** (u,i) *pair* **do**
6 Mine connected paths $P(u,i)$
7 // AKPM pre-train
8 **for** $t = 1; t \leq Iter; t++$ **do**
9 **foreach** u, v *pair* **do**
10 **for** $p_k \in P(u,i)$ **do**
11 $\mathbf{P}_k^h \leftarrow$ based on (2),(5), Combine($\mathbf{P^h}$) $\leftarrow$ $\mathbf{p_k^h}$;
12 $\mathbf{a}_{ui}^p \leftarrow$ attention pooling($Combine(\mathbf{P}^h)$) based on (4) and (5);
13 Calculate $\mathbf{\hat{y}_{ui}}$ based on (6);
14 Update AKPM parameters by back propagation through time;
15 // IFEM pre-train
16 **for** $t = 1; t \leq Iter; t++$ **do**
17 **foreach** u, v *pair* **do**
18 Calculate $\mathbf{\hat{y}_{ui}}$ based on (7) and (8);
19 Update IFEM parameters by back propagation through time;
20 Load pre-train AKPM parameters and IFEM parameters
21 **for** $t = 1; t \leq Iter; t++$ **do**
22 $\mathbf{a_{ui}^p} \leftarrow$ AKPM, $\mathbf{a_{ui}^I} \leftarrow$ IFEM;
23 Calculate $\hat{y}_{ui}$ based on (9);
24 Update EKPN parameters by back propagation through time;

Table 2 Dataset statistics

	dataset	IM-1M	Yelp
User-item interaction	Users	6040	37940
	Items	3382	11516
	Ratings	756684	229178
	Data density	3.70%	0.05%
Knowledge graph	Entities	18920	46606
	Entity types	11	7
	Links	800261	302937
	Link types	10	6
	Graph density	0.45%	0.03%

4 Experiments

In this section, we conduct experiments on two real datasets, MoviesLen-1M[1] and Yelp[2] to evaluate our method and compare the state-of-the-art KG-enhanced methods with our method to prove the effectiveness of our method. We also verified the importance of different components in the model through ablation experiments. The effectiveness of pre-training is finally analysed by comparing EKPN with and without pre-training.

4.1 Datasets

To prove the effectiveness of our recommendation model, we use two real-world datasets for verification. The first dataset is IM-1M, which is constructed by the combination of MovieLen-1M and IMDB[3]. The MovienLen-1M dataset is a movie rating dataset consisting of 3706 movies and 6040 users; IMDB contains auxiliary information for movies, including genre, actors, directors, etc. We link the two datasets by the title and release date of the movie and delete the unlinked movie data. After mapping MovienLen-1M and IMDB, we obtain 3382 movies and 6040 users, as well as 756,684 ratings. The second dataset is Yelp, which contains the user's evaluation of the local business and the business side information (such as location and category) (Table 2). According to the literature [24], we set the feedback that the user interacts with the item to 1; otherwise, we set it to 0.

4.2 Evaluation protocols

Leave-one-out has been widely used in prior efforts [22, 25, 26], so we use it to evaluate the recommendation performance for each dataset. Each user's latest interaction is used as the test set, and the remaining data are used as the training set. To reduce the complexity of testing, according to [15, 22], we randomly sample 100 items that the user does not interact with and then rank 101 test items for each user. We use the hit ratio (hit) and normalized discounted cumulative gain (NDCG) [27], which are widely used in previous efforts to evaluate the performance of top-K recommendations. Hit@K measures whether no not the top K positions of the recommendation list retrieve the test item or not. NDCG@K measures the relative order of the positive and negative items in the top K items in the ranking list.

4.3 Parameter settings

In the main experiments, we set the batch size to 256 and the learning rate to 0.001. For AKPM, we set the dimension of the relation and entity type to 64, set the dimension of entity and BiLSTM hidden size to 128, and set the dimension of $\mathbf{W}_q$,$\mathbf{W}_k$ and $\mathbf{W}_v$ size to 128. For IFEM, we set the dimension of the item to 128 and set the dimension of the fully connected layer size to 128. To prevent overfitting, we set the L2 regularization coefficient to 0.001. In addition, we use the Adam optimizer [28] to minimize the loss and the default Xavier initializer [29] to initialize the model parameters and the embedding layer.

4.4 Comparative methods

To demonstrate the effectiveness of our proposed model, we compare our proposed EKPN with the existing models:

- **FM** [1]: This method is a basic model that considers the combination of features and solves the problem of feature combination under sparse data;
- **NFM** [30]: This method combines a linear factorization machine with a nonlinear neural network to represent sparse data. Because the model can extract linear features and nonlinear features, state-of-the-art performance is achieved;
- **CKE** [5]: This is a representative embedding-based method that uses TransR [6] to capture semantic information in KG;
- **CFKG** [31]: This method uses transE to capture the semantic information in KG and designs a soft matching algorithm to explain personalized recommendations;
- **KPRN** [15]: KPRN is a path-based model that learns the user's interest preferences by extracting paths and uses LSTM to model the paths automatically;
- **CKAN** [32]: This method explicitly encodes the collaborative signals that are latent in the user-item interactions and naturally combines them with knowledge associations in an end-to-end manner.

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

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

[3] http://www.imdb.com/.

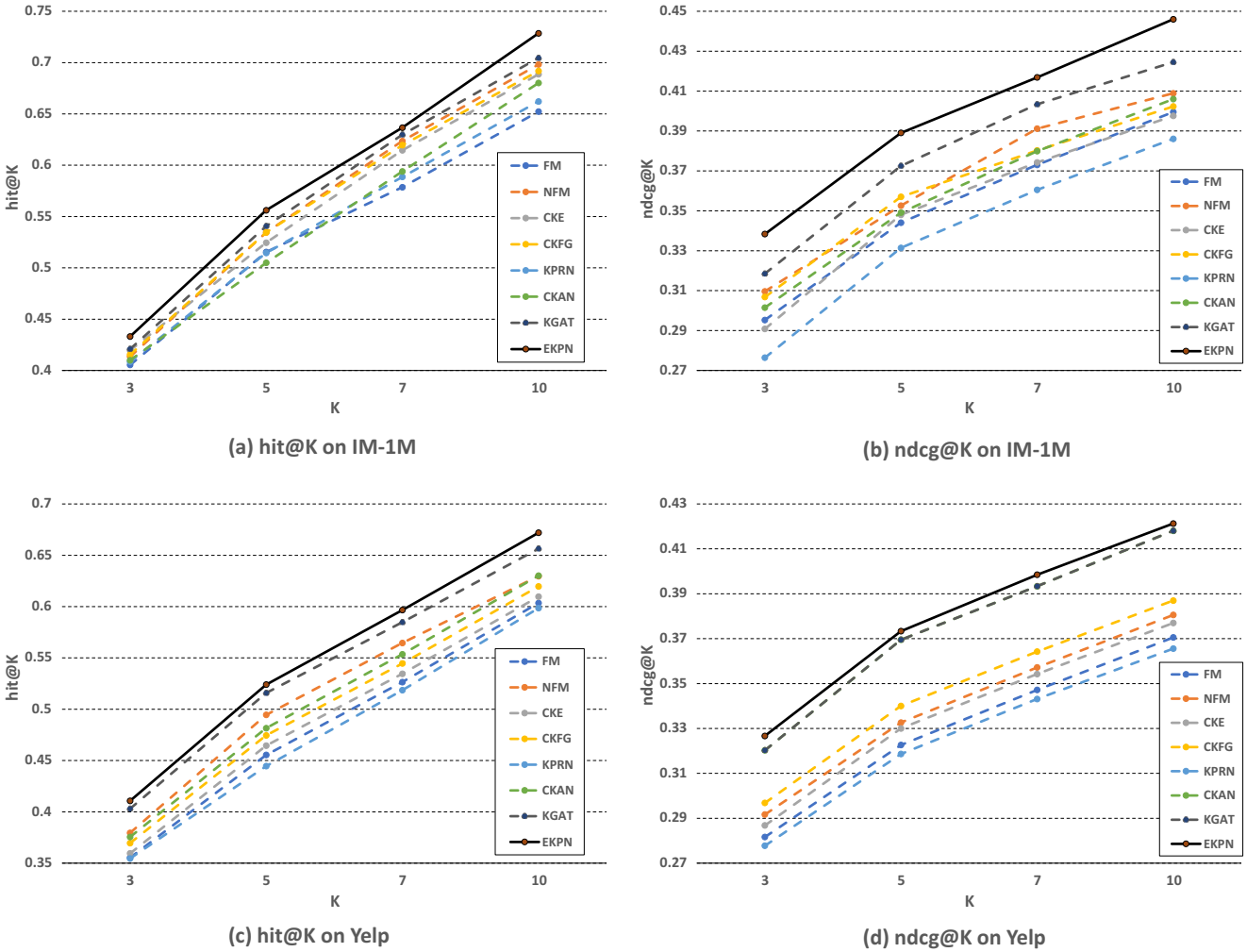

Fig. 5 The performance of all methods on the IM-1M and Yelp datasets Hit@k and NDCG@K when K is {3, 5, 7, 10}

– **KGAT** [11]: This method is an embedding-based method that uses GCN to explore the high-order connectivities in KG;

4.5 Performance comparison

Fig. 5 and Table 3 show the performance comparison results of each model. We observe the followings:

– From Fig. 5, we observed that our proposed model substantially outperforms KGAT Hit@K and NDCG@K, achieving the best performance. An examination of Table 3 shows that our model achieves the best performance and outperforms the previous best results with a 2.15% improvement on Hit@10 and a 2.25% improvement on NDCG@10 in the IM-1M dataset. In another Yelp dataset, we achieve a 1.56% improvement on Hit@10 and a 0.32% improvement on NDCG@10. These results demonstrate the effectiveness of our proposed model. Compared with KGAT and KPRN, we can conclude that our proposed IFEM can extract implicit features between items outside the knowledge graph.

Table 3 Overall performance comparison

Method	IM-1M		Yelp	
	HR@10	NDCG@10	HR@10	NDCG@10
FM	0.6685	0.3995	0.6035	0.3706
NFM	0.6982	0.4089	0.6396	0.3912
CKE	0.6626	0.3976	0.6096	0.3770
CKFG	0.6918	0.4023	0.6296	0.3852
KPRN	0.6621	0.3861	0.5985	0.3686
CKAN	0.6801	0.4060	0.6301	0.3925
KGAT	0.7045	0.4245	0.6564	0.4181
EKPN	**0.7258**	**0.4460**	**0.6720**	**0.4213**

[1] The improvement of our model over all baselines is statistically significant with $p < 0.01$ under paired t-test

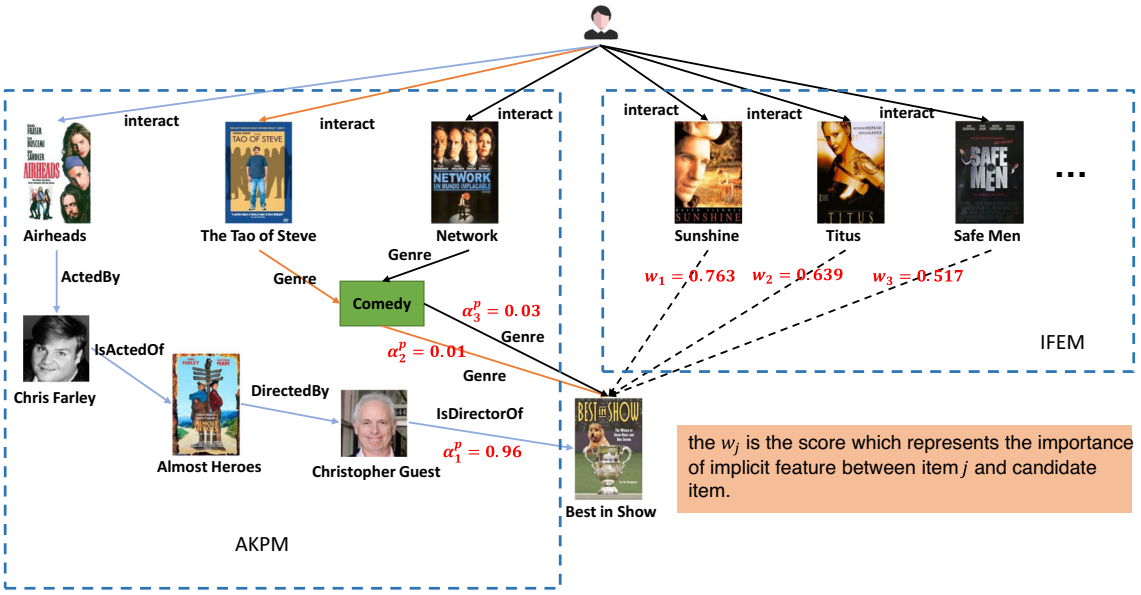

Fig. 6 A running example to help illustrate the interpretability of EKPN on the MovieLens-1M dataset. The $\boldsymbol{\alpha}_i^p$ is the i-th path score in (5), which uses $\mathbf{p_k^h}$ as input, and w_j is the score in (7), which represents the score between item j and candidate item

– In most cases, the performance of FM and NFM is better than those of CFKG and CKE, which indicates that the methods based on knowledge graph embedding may not be able to make full use of the knowledge graph. Although the performance characteristics of FM and NFM are better than those of CKE and CFKG, the performance in the latest model KGAT that uses a GAN network to capture the semantic features in the knowledge graph, is better than FM and NFM. However, KGAT extracts explicit features in the knowledge graph, so that its performance is weaker than that for our method. Through the comparison between Yelp dataset and IM-1M dataset, we find that our method has less improvement effect for Yelp dataset, which indicate that our method is data-dependent, and if there is less data, the recommendation performance will be relatively poor.

4.6 Case study

One function of EKPN is to capture features outside the knowledge graph. To demonstrate this functionality, we show an example drawn from EKPN on a movie recommendation task. We randomly selected a user whose ID is u3470 in IM-1M as an example. As shown in Fig. 6, the left part is an example using AKPM to model the path, and the right part is the score obtained between the candidate item and the item of the user historical behaviour by IFEM. We obtain the following results:

– This case demonstrates EKPN's capacity to provide informative explanations. By analysing these three paths, we can find different paths describing user and item connectivity from dissimilar angles, which can be used as evidence for why the candidate item is recommended. We can also provide some reasonable explanations where α_j^p represents the importance of path j.
– In addition, we used IFEM to display the scores between the candidate items and user historical behaviour items and only showed the items that had higher scores. We unexpectedly found that both "Sunshine" and "Best In Show" were awarded by the 58th Golden Globe Award nominations. Therefore, implicit features outside the knowledge graph exist

Table 4 Ablation comparison of our proposed model

Method	IM-1M		Yelp	
	HR@10	NDCG@10	HR@10	NDCG@10
w/o IFEM	0.6825	0.4084	0.5992	0.3896
w/o AKPM	0.7125	0.4321	0.6510	0.4043
w/o activation gate mechanism	0.6942	0.4160	0.6553	0.4103
Full model	**0.7258**	**0.4460**	**0.6720**	**0.4213**

Table 5 Performance of EKPN with/without pre-training

Datasets	Without pre-training		With pre-training	
	HR@10	NDCG@10	HR@10	NDCG@10
IM-1M	0.7194	0.4376	0.7258	0.4460
Yelp	0.6648	0.4135	0.6720	0.4213

between items, and the IFEM module can capture this implicit feature.

4.7 Ablation study

To investigate the effectiveness of EKPN, we conducted an ablation study to evaluate whether each module in our model contributes to our full model. We ablate three important components and conduct different approaches in this experiment.

- w/o IFEM, which is the model without IFEM that uses AKPM only to capture the explicit features between items in the knowledge graph for recommendation.
- w/o AKPM, which is the model without AKPM and only uses IFEM to explore implicit features between items outside the knowledge graph by utilizing the users' historical interactions.
- w/o activation gate mechanism, where is the model without the activation gate mechanism and only uses the average pool layer to transform the users' historical interactions into feature vectors.

Table 4 shows the performance of our model on the IM-1M and Yelp datasets by removing one module at a time. We find that all the components are useful for recommendation. From the table, we can see that the model that removed AKPM drops by 1.33% in Hit@10 and 1.39% in NDCG@10 on the IM-1M dataset and drops by 2.10% in Hit@10 and 1.70% in NDCG@10 on the Yelp dataset. If we remove the IFEM from the full model, the performance drops dramatically. The table shows that the model without IFEM drops by 4.21% in Hit@10 and 3.76% in NDCG@10 on the IM-1M dataset and drops by 7.28% in Hit@10 and 3.17% in NDCG@10 on the Yelp dataset. These results prove that IFEM can explore implicit features between items outside the knowledge graph by utilizing users' historical interactions. If we remove the activation gate mechanism and use only the average pool layer to transform the users' historical interactions into feature vectors, the performance drops dramatically. The table shows that the performance of the model without the activation gate mechanism drops by 3.16% in Hit@10 and 3.00% in NDCG@10 on the IM-1M dataset and drops by 1.67% in Hit@10 and 1.10% in NDCG@10 on the Yelp dataset. These results prove that the activation gate mechanism can better capture the implicit features between items outside the knowledge graph.

4.8 Impact of pre-training

We use two methods to initialize the parameters: one uses pre-training weights to initialize the EKPN, and the other uses the Xavier initializer [29] to initialize the model parameters. As shown in Table 5, the EKPN with pre-training outperforms that without pre-training in all cases. This result verifies that the pre-training process can ensure that IFEM and AKPM learn the features from different perspectives and reduce their mutual influence. Therefore, the pre-training process allows the model to generate better results.

5 Conclusions and future work

In this work, we seek to design a network that extracts explicit features between items within the knowledge graph and implicit features between items outside the knowledge graph. This approach solves the problem that existing knowledge graph-based methods find it difficult to capture the implicit features between items outside the knowledge graph. The effectiveness of our network was proven through extensive experiments.

In the future, we consider reflecting the implicit features between items by using embedding to solve this problem. In addition, the performance of path-based methods depends on the path selection method, but current path selection methods do not optimize recommendations. Therefore, we consider designing a model to select a path for recommendation.

Acknowledgments This work was supported in part by the Consulting Project of Chinese Academy of Engineering under Grant 2020-XY-5, 2018-XY-07, and in part by the Fundamental Research Funds for the Central Universities under Grant 2242021S30009, the Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

1. Rendle S, Gantner Z, Freudenthaler C, Schmidt-Thieme L (2011) Fast context-aware recommendations with factorization machines.

In: Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp 635–644
2. Cheng H-T, Koc L, Harmsen J, Shaked T, Chandra T, Aradhye H, Anderson G, Corrado G, Chai W, Ispir M, Anil R, Haque Z, Hong L, Jain V, Liu X, Shah H (2016) Wide & deep learning for recommender systems. In: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, pp 7–10
3. Guo H, Tang R, Ye Y, Li Z, He X (2017) Deepfm: A factorization-machine based neural network for CTR prediction. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, pp 1725–1731
4. Lian J, Zhou X, Zhang F, Chen Z, Xie X, Sun G (2018) xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp 1754–1763
5. Zhang F, Yuan NJ, Lian D, Xie X, Ma W-Y (2016) Collaborative knowledge base embedding for recommender systems. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, pp 353–362
6. Lin Y, Liu Z, Sun M, Liu Y, Zhu X (2015) Learning entity and relation embeddings for knowledge graph completion. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, pp 2181–2187
7. Wang H, Zhang F, Wang J, Zhao M, Li W, Xie X, Guo M (2018) Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp 417–426
8. Wang H, Zhao M, Xie X, Li W, Guo M (2019) Knowledge graph convolutional networks for recommender systems. In: The World Wide Web Conference, pp 3307–3313
9. Wang H, Zhang F, Zhang M, Leskovec J, Zhao M, Li W, Wang Z (2019) Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp 968–977
10. Hamilton WL, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, pp 1024–1034
11. Wang X, He X, Cao Y, Liu M, Chua T-S (2019) KGAT: knowledge graph attention network for recommendation. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp 950–958
12. Velickovic P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y (2018) Graph attention networks. In: 6th International Conference on Learning Representations
13. Shi C, Zhang Z, Luo P, Yu PS, Yue Y, Wu B (2015) Semantic path based personalized recommendation on weighted heterogeneous information networks. In: Proceedings of the 24th ACM International Conference on Information and Knowledge Management, pp 453–462
14. Sun Z, Yang J, Zhang J, Bozzon A, Huang L-K, Xu C (2018) Recurrent knowledge graph embedding for effective recommendation. In: Proceedings of the 12th ACM Conference on Recommender Systems, pp 297–305
15. Wang X, Wang D, Xu C, He X, Cao Y, Chua T-S (2019) Explainable reasoning over knowledge graphs for recommendation. In: The Thirty-Third AAAI Conference on Artificial Intelligence, pp 5329–5336
16. Xue H-J, Dai X, Zhang J, Huang S, Chen J (2017) Deep matrix factorization models for recommender systems. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, pp 3203–3209
17. Zhang L, Liu P, Gulla JA (2018) A deep joint network for session-based news recommendations with contextual augmentation. In: Proceedings of the 29th on Hypertext and Social Media, pp 201–209
18. Morales GDF, Gionis A, Lucchese C (2012) From chatter to headlines: harnessing the real-time web for personalized news recommendation. In: Proceedings of the Fifth International Conference on Web Search and Web Data Mining, pp 153–162
19. Bordes A, Usunier N, García-Durán A, Weston J, Yakhnenko O (2013) Translating embeddings for modeling multi-relational data. In: Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013, pp 2787–2795
20. Chaudhari S, Azaria A, Mitchell TM (2017) An entity graph based recommender system. AI Commun 30(2):141–149
21. Sun Y, Han J, Yan X, Yu PS, Wu T (2011) Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. Proc VLDB Endow 4(11):992–1003
22. He X, Liao L, Zhang H, Nie L, Hu X, Chua T-S (2017) Neural collaborative filtering. In: Proceedings of the 26th International Conference on World Wide Web, pp 173–182
23. Erhan D, Bengio Y, Courville AC, Manzagol P-A, Vincent P, Bengio S (2010) Why does unsupervised pre-training help deep learning? J Mach Learn Res 11:625–660
24. Catherine R, Cohen WW (2016) Personalized recommendations using knowledge graphs: A probabilistic logic programming approach. In: Proceedings of the 10th ACM Conference on Recommender Systems, pp 325–332
25. Bayer I, He X, Kanagal B, Rendle S (2017) A generic coordinate descent framework for learning from implicit feedback. In: Proceedings of the 26th International Conference on World Wide Web, pp 1341–1350
26. He X, Zhang H, Kan M-Y, Chua T-S (2016) Fast matrix factorization for online recommendation with implicit feedback. In: Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, pp 549–558
27. Järvelin K, Kekäläinen J (2017) IR evaluation methods for retrieving highly relevant documents. SIGIR Forum 51(2):243–250
28. Kingma DP, Ba J (2015) Adam: A method for stochastic optimization. In: 3rd International Conference on Learning Representations
29. Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Vol. 9 of JMLR Proceedings, pp 249–256
30. He X, Chua T-S (2017) Neural factorization machines for sparse predictive analytics. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp 355–364
31. Ai Q, Azizi V, Chen X, Zhang Y (2018) Learning heterogeneous knowledge base embeddings for explainable recommendation. Algorithms 11(9):137
32. Wang Z, Lin G, Tan H, Chen Q, Liu X (2020) CKAN: collaborative knowledge-aware attentive network for recommender systems. In: Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval. ACM, pp 219–228