KEP-Rec: A Knowledge Enhanced User-Item Relation Prediction Model for Personalized Recommendation

Lisha Wu, Daling Wang(✉), Shi Feng, Yifei Zhang, and Ge Yu

School of Computer Science and Engineering, Northeastern University, Shenyang, China
{wangdaling,fengshi,zhangyifei,yuge}@cse.neu.edu.cn

Abstract. For more accurate, diversified and interpretable personalized recommendation, the joint consideration of user-item interaction information and side information in knowledge graph has become a research hotspot. Traditional models based on collaborative filtering usually have cold start and sparse problems. The existing recommendation model based on knowledge graph can enrich the representation of users and items by using graph structure information from the knowledge graph, and make it more interpretable. Although the efforts have achieved a certain performance improvement, they consider all entities in knowledge graph globally for all users, and the aggregation strategy is single. In this paper, we propose KEP-Rec, a **K**nowledge **E**nhanced User-Item Relation **P**rediction Model for Personalized **Rec**ommendation. For a given target user and candidate item, KEP-Rec represents the user and item with enhanced information by knowledge graph for predicting the interacted probability between them and further personalized recommendation. In detail, KEP-Rec takes into account the changes in preferences of specific users and the differences in user perception of relations. Based on the idea of collaborative filtering, KEP-Rec selects an extended entity set of the items relevant with target user and candidate item as the initial set to propagate in knowledge graph. Moreover, KEP-Rec sets an item-aware attention mechanism to consider the interaction of candidate items with different weights given by target user's historical preferences to realize the diverse representation of the user preferences. In the propagation process alone knowledge graph, the relation embedding is considered for target user to achieve personalization. Empirical results on three real datasets of music, books, and movies show that KEP-Rec significantly outperforms state-of-the-art methods.

Keywords: User-Item Relation Prediction · Personalized recommendation · User and item embedding · Knowledge graph · Entity propagation

1 Introduction

At present, the recommender system (RS) becomes an effective approach to solve how users can efficiently obtain items they are interested in under the situation of "information overload". An effective traditional recommendation method is collaborative filtering (CF), which represents users and items as vectors, and models the historical interactions

B. Li et al. (Eds.): APWeb-WAIM 2022, LNCS 13422, pp. 239–254, 2023.
https://doi.org/10.1007/978-3-031-25198-6_19

between users and items as a matrix through operations such as inner product or neural network. However, CF-based models generally suffer from cold start and sparsity issues. In order to solve these problems, using knowledge graph as auxiliary information into the recommender system has attracted researchers. Knowledge graph (KG) is a heterogeneous graph, in which nodes are entities and edges represent relations between entities. The items and their attributes in the recommender system can be mapped to KG's entities which makes it easy to learn more about the relationship between items. Integrating user information into KG can capture more diverse user preferences. The existing KG-based recommender system models can be divided into embedding-based, path-based, and comprehensive methods [5]. The embedding-based approach enriches the item or user's presentation by directly using KG, but it ignores the connectivity of entities in KG. The path-based algorithm is used to explore multiple meta-paths between user and item in KG to infer the user preferences, but it will cost highly for artificially setting paths, and the rich structural information stored in KG is ignored. Both of them cannot mine and utilize well enough the comprehensive correlation between user and item. So comprehensive methods based on GNN have been proposed. For example, ripple2vec [25] proposes to implement node embedding by constructing a context graph via a new defined ripple distance over ripple vectors. GNN can directly model the high-order connectivity between entities. This high-order connection contains rich semantics that can refine the entity representation by leveraging the entity's multi-hop neighbor information.

Although these methods have improved interpretability and performance, they still have three limitations:

(1) Incorporating knowledge graph can contain more information, but there are fewer items that can align entities in the knowledge graph due to the user-item interaction sparse data. so that is not enough to mine more user-preferred entities.
(2) Directly using the items interacted by a user and mapping them to the knowledge graph can obtain more knowledge information, but this information is based on the user's historical behavior. In fact, a user's preference may be changeable. How to dynamically measure a user's diversified preference based on his history for the current candidate item is the key to recommender systems.
(3) Using GNN can refine the entity representation, which shows that using neighborhood information can greatly promote the completion of recommendation tasks. However, this method ignores that different users have different weight for the same relation in the graph. This weight can also be understood as different users' different cognition of entities and relations in the knowledge graph.

In order to solve the above problems, we propose KEP-Rec, a **K**nowledge **E**nhanced User-Item Relation **P**rediction Model for Personalized **Rec**ommendation that considers preference changes and collaborative interaction for specific users. For a given target user and candidate item, KEP-Rec represents the user and item by knowledge graph for predicting the interacted probability between them and further personalized recommendation. Similar with CKAN [18], KEP-Rec analyzes the items related with target

user and the users related with candidate item and more relevant items. However, KEP-Rec considers the influence of candidate items on user preferences and pays different attention to the relation of the target users in KG by two attention mechanisms.

In general, our contributions are the following:

(1) We propose KEP-Rec model, an end-to-end collaborative knowledge propagation user-item representation framework for user-item relation predicting and further personalized recommendation.
(2) We construct a diversified representation of user preferences, that is, the impact of candidate items on users' historical preferences has changed the user's preferences. Moreover, the user-specific relation perception can obtain the preference of the target user to the knowledge relations.
(3) We conducted empirical experiments in three real-world recommendation scenarios datasets, and the results showed that our KEP-Rec is better than the existing state-of-the-art baselines.

2 Related Work

Now, recommender systems based on KG have been implemented in three ways: embedding-based methods, path-based methods and comprehensive methods [5].

2.1 Embedding-Based Recommendation

This method directly encodes entities into low-rank embeddings and uses the semantic information of the user/item in the KG for recommendation. The algorithm models can be divided commonly into two categories: translation distance models, such as TransE [1], TransH [19], TransR [9], and semantic matching models, such as DistMult [20]. The classic CKE [22] model integrates various auxiliary information in the framework of collaborative filtering which uses the TransR algorithm to encode item's structured knowledge and integrates the content knowledge to represent the item. Another model called CFKG [23] constructs a user-item knowledge graph. In that KG, user, item and their related attributes are regarded as entities, and the users' historical behaviors are regarded as a special type of relationship between entities. Based on a specific distance function, the model can learn user/item embeddings which is the implementation of semantic matching models.

2.2 Path-Based Recommendation

The common path-based method calculates the semantic similarity between entities on different paths to make recommendation. In HeteRec [21], L different types of meta-paths connecting users and items are defined. The similarity of items in each path is measured by PathSim [12] and formed L item extended matrices. Because the interaction matrices between users and items, multiplying with the item extended matrices can get L extended interaction matrices. Finally, the non-negative matrix factorization technique [3] is applied to obtain the latent vectors of users and items in different meta-paths which enriches users' and items' representations.

2.3 Comprehensive Methods

Both embedding-based and path-based methods only use partial information in knowledge graph. In order to make better use of knowledge graph, a comprehensive method is proposed to extend the embedding of the entity in the graph, directly model the high-order connectivity between entities, and use the entity's multi-hop neighbor information to refine the entity representation. RippleNet [15] is the first to propose the concept of preference propagation, which enriches user representations by aggregating the multi-hop neighbors of the user's historical interactive items. KGCN [16] is to aggregate the candidate item with its multi-hop neighbors to update the item's representation. In the aggregation process, the weight of the neighbor is jointly determined by the user and the candidate item, so that the user's preferences are implicitly existing in the entity representation. Another model, such as KGAT [17], combines user-item interaction matrix with a knowledge graph containing attribute information and uniformly represents users and items in the same graph. Then users and items are aggregated with their respective multi-hop neighbors in the graph to enrich their representations. Recently, CKAN [18] uses a heterogeneous propagation strategy to explicitly encode collaborative signals and knowledge associations and applies a knowledge-aware attention mechanism to distinguish the contributions of different neighbors.

Although the above methods achieve a certain performance improvement, they consider all entities in the knowledge graph at a global level and the aggregation strategy is single. Inspired by GARG [26], which takes full advantage of the collaborative, sequential and content-aware information, we first use collaborative information to expand the initial entity set of users and items. In particular, we think that for a specific target user, the relation between users, items, and entities in KG is not fully equal, so the KEP-Rec model considers the influence of candidate items on user preferences and pays different attention to the relation of the target users in KG.

3 Problem Definition

There are a set of M users and a set of N items are expressed as $U = \{u_1, u_2, \ldots, u_M\}$ and $V = \{v_1, v_2, \ldots, v_N\}$ respectively in a typical recommender system. We define the user-item interaction as a binary matrix $Y = \{y_{uv} | u \in U, v \in V\}$, where $y_{uv} = 1$ means that the user u ever interacted with the item v, such as clicking, collecting or purchasing; otherwise $y_{uv} = 0$. Note, the value of 1 for y_{uv} indicates that there is an explicit interaction between the user u and the item v, but doesn't necessarily mean u's preference over v.

In addition, we have a knowledge graph $\mathcal{G} = \{(h, r, t) | h, t \in \mathcal{E}, r \in \mathcal{R}\}$ which consists of massive knowledge triples. Each triple (h, r, t) demonstrates that a relation r exists between head entity h and tail entity t. And the sets of entities and relations in $\mathcal{G}$ are denoted as $\mathcal{E}$ and $\mathcal{R}$. It is also mentioned in [27] that representing machine-interpretable statements in the form of subject-predicate-object triples is a mature practice for capturing the semantics of structured data. For example, the triple (*A Song of Ice and Fire, book.book.author, George Martin*) states the fact that George Martin writes the book "A Song of Ice and Fire". In this triple, *A Song of Ice and Fire* is head entity, *book.book.author* is relation, and *George Martin* is tail entity.

The problem we want to solve is defined as follows: given a knowledge graph $\mathcal{G}$ and the historical interaction matrix Y between user set U and item set V, for a target user $u_T \in U$ and a candidate item $v_C \in V$, predict the probability that u_T would interacts with v_C which has not interacted with before, i.e. $y_{u_T v_C} = 0$ in Y.

For solving above problem, we apply knowledge graph $\mathcal{G}_{\mathcal{N}}$, assume that every item $v_j \in V$ $(j = 1, 2, \ldots, N)$ in interaction matrix Y can be linked to a corresponding entity $e \in \mathcal{G}$ by the entity linking technique [2]. Moreover, based on U, V, Y, and $\mathcal{G}_{\mathcal{N}}$, we learn a prediction function $\hat{y}_{uv} = f(u_T, v_C | \Theta, Y, \mathcal{G})_{\mathcal{N}}$, where $\hat{y}_{uv}$ represents the probability of our prediction, and Θ represents the parameters of the function.

4 Methodology

Our proposed KEP-Rec framework is shown as Fig. 1, the model contains three main layers, we introduce them in detail in Sect. 4.1, 4.2, and 4.3 respectively.

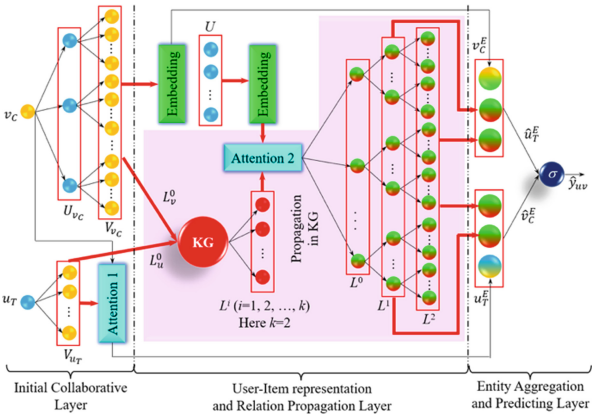

Fig. 1. The framework of KEP-Rec, where the light blue dot denotes user, the orange dot denotes item, the red dot denotes the entity in KG, and the layered color dots are the different embedding representations respectively. In the pink background, it is emphasized that the two initial entity sets are based on the relation-aware propagation process in KG, where L^i $(i = 1, 2, \ldots, k)$ represents the entity set after each layer of propagation.

4.1 Initial Collaborative Layer

The layer is shown as the left side in Fig. 1 and consists of three parts. The first part is to construct the initial entity set of target user u_T, the second part is to construct the initial entity set of candidate item v_C, and the third part is to consider the influence of v_C on the preference of u_T. Next, we will explain these three parts in detail.

The Target User's Initial Entity Set. Intuitively, the user's preferences can be reflected through the items which have interacted with, so we can consider these items as the user's

initial entity set to express user preferences. From the user's point of view, starting from the entities in this initial entity set and propagating along the connections in the knowledge graph can reasonably expand the user's preference range and enrich the user's representation. For target user u_T, his interacted history with items is V_{u_T}, where $V_{u_T} \subset V, \forall v \in V_{u_T}$ satisfies $y_{u_r v} = 1$ in Y and the set size of V_{u_T} denoted as $m = \left|V_{u_T}\right|$. We define V_{u_T} as the initial entity set of target user u_T.

The Candidate-Item Initial Entity Set. User-based collaborative filtering [14] assumes that users who like the same items are similar. According to this assumption, we can consider that users who ever interacted with candidate items v_C have similar preferences. So, it is a great extension to use these items that the users have interacted with the initial entity set of v_C. As shown in the upper left of Fig. 1, we denote the user set interacted with v_C as U_{V_C} where $U_{V_C} \subset U, \forall u \in U_{V_c}$ satisfies $y_{uv_C} = 1$ in Y. Then, for every $u \in U_{V_C}$, we denote the item set interacted with u as V_u, where $V_u \subset V, \forall v \in V_u$ satisfies $y_{uv} = 1$ in Y. And then $\forall u \in U_{V_C}$, we take the union of all entity sets V_u constructed by u denoted as V_{v_C} and the set size of V_{v_C} denoted as $n = \left|V_{v_c}\right|$. Equally, for every $v = V_{v_C}$, the v interacted with at least a $u \in V_{V_c}$. Finally, we define V_{v_C} as initial entity set of candidate item v_C. In this case, v^E is the origin embedding of $v \in V$, and the candidate item v_C initial entity embedding is directly represented as:

$$v_C^E = \frac{\sum_{j=1}^{n} v_j^E}{n}, v_j \in V_{v_c} \tag{1}$$

The Item-Aware Attention Mechanism [13]. In fact, for all $v_i \in V_{v_C}$ $(i = 1, 2, \ldots, n)$ the items can only reflect the historical preferences of $u \in U_{V_C}$. As mentioned in [27], the existing recommendation methods mainly consider the representation of users as static feature sets. But user preferences are diverse and changeable, and the newly candidate items v_C may affect user preferences. For example, user A has watched *Avengers* and *Iron Man*, both of which belong to action movies. Then the user watching the romance movie will affect his preference for action movies. To characterize user's diverse interests, we use an item-aware attention network, i.e. Attention 1 in Fig. 1, to model the different impacts of the user's historical interaction items on the candidate item. Specifically, we apply a multi-layer perceptron to calculate the score between candidate item and historical items and SoftMax [10] function to calculate the normalized impact weight:

$$\pi(v_C \| v_i) = W_2(ReLu(W_1(v_C \| v_i) + b_1)) + b_2 \tag{2}$$

$$\pi(v_C \| v_i) = softmax\left(\frac{\pi(v_C \| v_i)}{\sum_{i=1}^{m} \pi(v_C \| v_i)}\right) \tag{3}$$

where $v_i \in V_{u_T}$ and $m = \left|V_{u_T}\right|$. Finally, we get the weighted initials user embedding:

$$u_T^E = \sum_{i=1}^{m} \pi(v_C \| v_i) * v_i \tag{4}$$

Through this item-aware attention mechanism to aggregate user historical interactive items, it is possible to realize the diverse user preferences, which better represents real-life scenarios in the selection process of items by users.

4.2 User-Item Representation and Relation Propagation Layer

The layer is shown as the middle side in Fig. 1. In this layer, we mainly accomplish two tasks: one is knowledge graph propagation, that is, start from the initial entity set V_{u_T} and V_{v_C} we defined above, and use relation links to obtain extended entities and triples; the other is constructing a user-specific relation attention mechanism.

For introducing knowledge graph $\mathcal{G}$, we define $L_u^0 = V_{u_T}$ and $L_v^0 = V_{v_C}$ as 0^{th} layer entity set in $\mathcal{G}$.. Based on the graph structure of $\mathcal{G}$, the entities L_u^0 and L_v^0 can be iteratively propagated along relations to reach more connected entities and we define the i^{th} ($i = 1, 2, \ldots, k$, where k is the layer number propagated finally) set of entities as follows:

$$L^i = \{t^i | (h^{i-1}, r^i, t^i) \in \mathcal{G}, h^{i-1} \in L^{i-1}, with\ i = 1, 2, \ldots, k \tag{5}$$

After k hops propagation, we have $k + 1$ sets of entity sets L^i ($i = 1, 2, \ldots, k$).

The traditional graph convolutional network only propagates the information embedded by the entity itself and ignores the semantic information encoded in the relationship. In KG, an entity can be connected to multiple neighbors through various relationships, indicating different semantics. In our KEP-Rec model, the user-oriented personalized recommendation is embodied in that for different users, the KG entity should have different representations to characterize its embedding. For example, for the movie entity, some users may watch the movie *Iron Man* because the actor Robert is famous. Others watch the movie *The Fantasy Drifting of the Youth Pie* because the director is famous by Ang Lee, so the rich semantics encoded in the relationship is crucial to understanding different user intentions.

In G, we need to consider the entity feature representation when facing different users, that is, setting different weights for tail entities to reveal the different semantic information of different head entities and relations. Specifically, for a given entity, neighbors under each relation surrounding the entity are scored in a user-specific function that measures the influence of each entity's neighbors. After k^{th} layers propagations, we can get k^{th} set of tail entities. And we set a tail entity's embedding t_i^l with user-specific attentive weight as such:

$$t_i^l = A(h^l, r^l, u_{origin}^E) t_i^l \tag{6}$$

where h^l is the head entity's embedding, r^l is the relation's embedding, t^l is the tail entity's embedding of the l^{th} triple, and u_{origin}^E is the user's origin embedding. $A(h^l, r^l, u_{origin}^E)$ is the attentive weight impacted by the head entity, the relation between head h^l and tail r^l and specific user u_{origin}^E which implemented by the two-layer neural network (see Attention 2 in Fig. 1):

$$a_0 = ReLu\Big(W_0\Big(h^l \| r^l \| u_{origin}^E\Big) + b_0\Big) \tag{7}$$

$$A(h^l, r^l, u_{origin}^E) = softmax(\sigma(ReLu(W_1 a_0 + b_1)) \tag{8}$$

Here we chose ReLU [6] and Sigmoid [7] as the activation functions. W and b are trainable weight matrices and characteristic parameters, and their respective subscripts

indicate different layers. This user-specific relation attention mechanism uses the information from a given user, item, and relation to determine which neighbor has more information about the item while increasing personalized choices.

Based on the above, we use the initial entity set V_{u_T} and V_{V_C} for target user u_T and candidate item v_C respectively as the first head entity to be propagated in $\mathcal{G}$, then, the k^{th} layer entities set embedding can be represented as: $t^k = \sum t_i^k, t_i^k \in Lk$. After integrating all entities k layers propagation, we can generate the final entity embedding of $t = \{t^1, t^2, \ldots, t^k\}$.

Finally, we combine initial entity set representations of target user u_T and candidate item v_C initial entity set representations and the entity sets representations that have been propagated in $\mathcal{G}$to obtain user and item final representations as follow:

$$\hat{u}_T^E = \left\{u_T^E, t_u^1, t_u^2, \ldots, t_u^L\right\} \tag{9}$$

$$\hat{v}_C^E = \{v_C^E, t_v^1, t_v^2, \ldots, t_v^L\} \tag{10}$$

4.3 Entity Aggregation and Prediction Layer

The final step in this model is the prediction shown in the right side in Fig. 1, which aggregates entity itself representation and its multi-hop neighbors after k layers knowledge propagation. There are three common aggregators that we can aggregate multi-embeddings and we have implemented them in our model.

Sum Aggregator. The *Sum* aggregator sums multiple representations, followed by a nonlinear transformation:

$$agg_{sum} = \sigma(W \cdot (e_1 + e_2 + \cdots + e_n) + b) \tag{11}$$

Concat Aggregator. The *Concat* aggregator concatenates multiple representations, and then applies a nonlinear transformation:

$$agg_{concat} = \sigma(W \cdot (e_1 \| e_2 \| \ldots \| e_n) + b) \tag{12}$$

Pooling Aggregator. The *Pooling* aggregator takes the maximum value of multiple vectors as the same dimension, followed by a non-linear transformation:

$$agg_{pool} = \sigma(W \cdot maxpool(e_1, e_2, \ldots, e_n) + b) \tag{13}$$

According to our experimental results, we apply *Concat* Aggregator in KEP-Rec model. Based on the aggregators, we can further predict the interaction probability for target user u_T and candidate item v_C.

Model Prediction. Based on the target user' representation $\hat{u}_T^E$ and candidate item's representation $\hat{v}_C^E$, the predicted probability is calculated by $\hat{y}_{uv} = \sigma(\hat{u}_T^{E\mathrm{T}} \hat{v}_C^E)$, where $\sigma()$ is the sigmoid function.

Loss Function. For each user, we randomly select the same number of negative samples with positive samples to make sure the effectivity of model training. We define the loss function of the model KEP-Rec as follows:

$$\mathcal{L} = \sum_{u \in U} \left(\sum_{v \in \{v|(u,v) \in \mathrm{P}^+\}} \mathcal{J}\left(y_{uv}, \hat{y}_{uv}\right) - \sum_{v \in \{v|(u,v) \in \mathrm{P}^-\}} \mathcal{J}\left(y_{uv}, \hat{y}_{uv}\right) \right) + \lambda ||\Theta||_2^2 \tag{14}$$

where $\mathcal{J}$ is the cross-entropy loss, P^+ is the positive sample while P^- means the negative sample. Θ is the model parameter set, and $||\Theta||_2^2$ is the L_2-regularizer that is parameterized by λ.

5 Experiments

In this section, we evaluate the KEP-Rec model in three real-world scenario datasets. Inspired by mostly related work and the paper [24] which discussed the value of experimentation and measurement, the experiments will answer the following research questions:

Q1: How does KEP-Rec perform compared with the state-of-the-art KG-based recommendation methods?

Q2: How do different parameters affect KEP-Rec?

5.1 Datasets

In order to verify the effectiveness of KEP-Rec in different application scenarios, we apply three general used datasets from different fields (movies, books, and music) in our experiments.

- **MovieLens-20M**[1] **(ML for short)**: This dataset is collected by GroupLens Research which obtained nearly 20 million rating (from 1 to 5) from 27,000 movies by 138 thousand users on the MovieLens website.
- **Book-crossing**[2] **(BC for short):** This dataset is collected by Cai-Nicolas Ziegler from the Book-Crossing community (August to September 2004). It contains 278,858 users 1,149,780 ratings (from 1 to 10) for approximately 271,379 books.
- **Last.FM**[3] **(FM for short):** This dataset contains the social networks and music artist information of two thousand users who listened to the online music system of Last.FM.

Since KEP-Rec aims to predict the interacted probability between target user and candidate item and make recommendation based on implicit feedback, we set a scoring threshold to convert the explicit feedback into implicit feedback. For MovieLens-20M, we set the positive score threshold to 4, and scores greater than 4 are positive samples

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

[2] http://www2.informatik.uni-freiburg.de/.cziegler/BX/.

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

($y_{uv} = 1$ in Y), and vice versa ($y_{uv} = 0$ in Y). However, the Book-Crossing and Last.FM data are too sparse, and the above threshold settings are not suitable. So, for those two datasets, we set the items that the user interacts with as the positive samples. For negative samples, we randomly select items of the same size as the positive samples from items that the user has not interacted with.

In addition to the aforementioned user and item interaction datasets, we also need to choose sub-KGs of each dataset. For MovieLens-20M, Book-Crossing, and Last.FM, we choose sub-KGs from KG called Satori[4] from Microsoft. Each sub-KG is a subset of the entire KG, the confidence level is greater than 0.9. For the sake of simplicity, we exclude the items matched with multiple entities and those unmatched to any of entities. Table 1 summarizes the statistics of these experimental datasets.

Table 1. Statistics of Movie-Lens20M, Last.FM, and Book-Crossing, where avgI means the average interactions per user, and avgL means the average link per entity

	#users	#items	#interaction	#avgI	#entities	#relations	#triples	#avgL
ML	138,159	16,954	13,501,622	23	102,569	32	499,474	29
FM	1,872	3,846	42,346	98	9,366	60	15,518	4
BC	17,860	14,967	139,746	8	77,903	25	13,150	10

5.2 Baselines

In the experiments, we will compare our KEP-Rec model with the following baselines.

- **BPRMF** [11] uses matrix factorization based on Bayesian personalized ranking, which is based on the user's paired preferences as a single collaborative filtering method.
- **CKE** [22] combines a CF module with knowledge embedding, text embedding, and image embedding of items in a unified framework and jointly learn to make recommendations.
- **RippleNet** [15] is a state-of-the-art propagation-based model that uses a large number of entities related to the user's historical clicks to enrich the user's representation, so that the user's potential preferences can be propagated in KG. Then the click-through rate of the user-item pair is predicted.
- **KGAT** [17] uses embedded propagation to directly model high-connectivity between users and items. It applies TransR model to obtain the initial representation of the entity. Then it runs entity propagation from the entity itself along the relationship link in the knowledge graph. In the process of outward propagation, the information from the entity will iteratively interact with the multi-hop neighbor.
- **CKAN** [18] uses a heterogeneous propagation strategy to explicitly encode collaborative signals and knowledge associations, and applies a knowledge-aware attention mechanism to distinguish the contributions of different knowledge-based neighbors.

[4] https://searchengineland.com/library/bing/bing-satori.

5.3 Experimental Setup

In our experiment, for each dataset, 60% are randomly selected for training, 20% are for evaluation, and the rest 20% are for prediction. We evaluate our method in CTR prediction and top-K recommendation. CTR prediction generally refers to the click-through rate estimation task which is with an item, predicting the probability that the user clicks on this item. We adopt AUC to evaluate this performance. For top-K recommendation, we adopt F1@K to evaluate the performance. For optimization, we use ADAM [8] to optimize all models in training. We set the batch size to 1024 during training and use the default Xavier initialize [4] to initialize the parameters of the model.

We implement our model in PyTorch. The best hyper-parameters are obtained by grid search. We set the learning rate to be searched in $\{10^{-3}, 5 * 10^{-3}, 10^{-2}, 5 * 10^{-2}\}$. The embedding size is tuned among {16, 32, 64, 128, 256}. The coefficient of L_2 normalization is searched in $\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$. We search the set of size in {4, 8, 16, 32, 64} for user and item embedding.

5.4 Experimental Results and Analysis

The experimental results and analysis are shown as follows.

Performance Comparison with Baselines (Q1). In this section, we present the results of performance comparisons among KEP-Rec and baselines. The results of all methods in CTR prediction and top-K recommendation are presented in Table 2 respectively.

Table 2. The result of AUC and F1 in CTR prediction comparing of different models

Dataset	ML		FM		BC	
Model	AUC	F1	AUC	F1	AUC	F1
BPRFM	0.958 (−2.6%)	0.914 (−2.7%)	0.756 (−9.7%)	0.701 (−8.6%)	0.658 (−9.5%)	0.611 (−6.6%)
CKE	0.927 (−5.7%)	0.874 (−6.7%)	0.747 (−10.6%)	0.674 (−11.3%)	0.676 (−7.7%)	0.623 (−5.4%)
RippleNet	0.976 (−0.8%)	0.927 (−1.4%)	0.776 (−7.7%)	0.702 (−8.5%)	0.721 (−3.2%)	0.647 (−3.0%)
KGAT	0.976 (−0.8%)	0.928 (−1.3%)	0.829 (−2.4%)	0.742 (−4.5%)	0.731 (−2.2%)	0.654 (−2.3%)
CKAN	0.976 (−0.8%)	0.929 (−1.2%)	0.842 (−1.1%)	0.769 (−1.8%)	0.753 (−0%)	0.673 (−0.5%)
KEP-Rec	**0.984**	**0.941**	**0.853**	**0.787**	**0.753**	**0.677**

From Table 2, we can be observed that:

(1) KEP-Rec consistently outperforms all baselines across mostly datasets in terms of all measures. More specifically, it achieves significant improvements over the

strongest baseline CKAN w.r.t. F1 by 1.2%, 1.8%, and 0.5% in MovieLens-20M, Last.FM and Book-Crossing respectively. That may because the two attention mechanisms played a big role in KEP-Rec. It is worth mentioning that on the MovieLens-20M dataset, the results of all baselines are at a higher value. It shows that more user interaction is conducive to make a better recommendation.

(2) The two path-based baselines, RippleNet and KGAT, are better than the CF-based method BRPFM and the embedding-based CKE, indicating that KG's graph structure information is helpful for recommendation. In addition, although RippleNet and KGAT achieved excellent performance, they still did not exceed KEP-Rec. This is because RippleNet neither incorporates user click history items into the user representation, nor does it introduce high-level connections, and KGAT does not mix GCN layer information, nor does it consider user preferences when collecting KG information.

(3) From the experimental results, the method based on KG propagation is higher than the pure CF-based BPRMF model on the three data sets of all evaluation indicators. This experimental result shows that the use of KG is very helpful for recommendation. However, it is worth noting that BPRMF outperforms CKE in some indicators, which means that modeling direct relationship in KG may not be able to make full use of the rich information encoded in KG and proved the effectiveness of high-level connectivity information. The second reason may be that CKE is aimed at multi-modal information, and only one of the graph structure information is used here, which leads to its poor performance.

(4) In addition, by comparing CKAN, KEP-Rec and KGAT, although both utilize high-level connectivity, CKAN and KEP-Rec outperforms KGAT. The possible reason for the analysis is that while modeling the item presentation, CKAN and KEP-Rec both further considers the collaborative signals in the interaction between the user and the product, thereby realizing the enrichment of the item. They explore the connections between users, item and entities through collaborative interaction and knowledge graphs.

Performance Comparison with Different Parameters (Q2). To get deep insights on different parameters of KEP-Rec, we investigate their impact in three datasets respectively. We first study the influence of layer numbers, and then examine the influence of knowledge graph embedding dimension. Finally, we analyze the influence of item attention and user-specific relationship propagation layer.

Impact of Number of Layers: The impact of different number of layers is shown in Table 3. We conducted experiments with the same other parameter settings. The results show that when layer number is 2, ML, and BC perform best; when layer number reaches 4, FM achieves better performance.

According to the analysis of different datasets, one possible reason for this phenomenon is that when there are more links in the knowledge graph of the dataset, long-distance propagation provides more supplementary knowledge information, but also brings more noise. However, in the case of a small amount of data in the knowledge graph of the dataset, deeper propagation can make use of knowledge information to a greater extent.

Table 3. F1 result of KEP-Rec with the different number of layer number.

Layer number	2	3	4
ML	**0.937**	0.934	0.934
FM	0.783	0.781	**0.787**
BC	**0.677**	0.673	0.671

Impact of Dimension of Embedding. We use the same dimensional parameters to embed the entities and relations in the KG, and compare the performance of KEP-Rec on all three data sets for different dimensions. The result is shown as Table 4.

Table 4. F1 result of KEP-Rec with dimension of embedding.

Embedding dimension	32	64	128	256
ML	0.941	0.941	0.941	0.941
FM	0.775	0.784	0.785	0.783
BC	0.668	0.669	0.671	0.672

From Table 4, it can be seen that the dimensional changes of KEP-Rec on the three datasets did not cause excessive fluctuations in the final evaluation index. This means that it has a strong tolerance for size selection, which reduces the dependence of the experiment on parameters and makes it easier to reproduce the experimental results.

Impact of Different Personalized Components. In order to verify the influence of item attention and user-specific relation propagation layer, we conducted ablation experiments of two sub-models. One is KEP-Rec that only removes item attention and we mark it as KEP-Rec$_{(-A)}$, and the other is to remove two parts at the same time, and we mark it as KEP-Rec $_{(-U-A)}$. The results are shown in Table 5.

Table 5. F1 result of KEP-Rec without different components. KEP-Rec $_{(-U-A)}$ means KEP-Rec without item-aware attention mechanism and user-specific relation layer. KEP-Rec $_{(-A)}$ means without user-specific relation propagation layer.

	KEP-Rec $_{(-U-A)}$	KEP-Rec$_{(-A)}$	KEP-Rec
ML	0.929	0.934	**0.941**
FM	0.773	0.778	**0.787**
BC	0.674	0.675	**0.677**

The result supports that item attention and user-specific relation propagation layer are both powerful determinants, and the combination of the two can be more completely encoded into the potential user/item vector representation.

Impact of Aggregators. In order to verify the influence of the aggregator on the results of our model, we chose **Sum**, **Pool** and **Concat** three aggregators, while keeping other conditions consistent. The experimental results are shown in Table 6.

Table 6. F1 result of KEP-Rec with different aggregators.

	Sum	Pool	Concat
ML	0.934	0.924	**0.941**
FM	0.736	0.733	**0.787**
BC	0.666	0.654	**0.677**

By analyzing the experimental results, we have the following observations that *Concat* is always better than *Sum* and *Pool*. This may be because, compared with the other two aggregators, the *Concat* aggregator can retain the information content of the embedded representation as much as possible without filtering and mixing. Based on the result, *Concat* aggregator is applied in our KEP-Rec model.

6 Conclusion

In this work, we propose KEP-Rec, a knowledge enhanced user-item relation prediction for users' diverse preferences representation, which is an end-to-end, user-oriented, and collaborative knowledge propagation prediction model of user-item relation for personalized recommendation. We construct a diverse representation of user preferences and set a user-specific relation attention mechanism to describe those relations between different users and the same item. Meanwhile, the high-order connectivity of the knowledge graph is used to finally obtain an enhanced representation of users and items. A large number of experiments have proved the superiority of KEP-Rec.

Since the proposed personalized component is aimed at entities existing in knowledge graph, the method can also be applied to fields related to graph structure, such as social networks. We believe that KEP-Rec can be widely used in related applications. In addition, the user's preference is actually also changed by time, and a real-going idea is to use CNN to join the time series in the model which is a viable direction.

Acknowledgement. The work was supported by National Natural Science Foundation of China (62172086, 61872074, 62106039).

References

1. Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O.: Translating Embeddings for Modeling Multi-relational Data. NIPS 2013, pp. 2787–2795 (2013)
2. David, N., Ian, H.: Witten: learning to link with wikipedia. CIKM 2008, pp. 509–518 (2008)
3. Ding, C., Tao Li, T., Jordan, M.: Convex and semi-nonnegative matrix factorizations. IEEE Trans. Pattern Anal. Mach. Intell. **32**(1), 45–55 (2010)
4. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. AISTATS 2010, pp. 249–256 (2010)
5. Guo, Q., et al.: A Survey on Knowledge Graph-Based Recommender Systems. CoRR abs/2003.00911 (2020)
6. Hahnloser, R., Sarpeshkar, R., Mahowald, M.A., et al.: Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. Nature **405**(6789), 947–951 (2000)
7. Han, J., Moraga, C.: The influence of the sigmoid function parameters on the speed of backpropagation learning. In: Mira, J., Sandoval, F. (eds.) IWANN 1995. LNCS, vol. 930, pp. 195–201. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-59497-3_175
8. Kingma, D., Ba, J.: Adam: A Method for Stochastic Optimization. ICLR (Poster) (2015)
9. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning Entity and Relation Embeddings for Knowledge Graph Completion. AAAI 2015, pp. 2181–2187 (2015)
10. Memisevic, R., Zach, C., Hinton, G., Pollefeys, M.: Gated Softmax Classification. NIPS 2010: 1603–1611 (2010)
11. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian Personalized Ranking from Implicit Feedback. UAI 2009, pp. 452–461 (2009)
12. Sun, Y., Han, J., Yan, X., Yu, P., Wu, T.: PathSim: meta path-based Top-K similarity search in heterogeneous information networks. Proc. VLDB Endow. **4**(11), 992–1003 (2011)
13. Vaswani, A., et al.: Attention is All you Need. NIPS 2017, pp. 5998–6008 (2017)
14. Wang, C., Blei, D.: Collaborative topic modeling for recommending scientific articles. KDD 2011, pp. 448–456 (2011)
15. Wang, H., et al.: RippleNet: Propagating User Preferences on the Knowledge Graph for Recommender Systems. CIKM 2018, pp. 417–426 (2018)
16. Wang, H., Zhao, M., Xie, X., Li, W., Guo, M.: Knowledge Graph Convolutional Networks for Recommender Systems. WWW 2019, pp. 3307–3313 (2019)
17. Wang, X., He, X., Cao, Y., Liu, M., Chua, T.: KGAT: Knowledge Graph Attention Network for Recommendation. KDD 2019, pp. 950–958 (2019)
18. Wang, Z., Lin, G., Tan, H., Chen, Q., Liu, X.: CKAN: Collaborative Knowledge-aware Attentive Network for Recommender Systems. SIGIR 2020, pp. 219–228 (2020)
19. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge Graph Embedding by Translating on Hyperplanes. AAAI 2014, pp. 1112–1119 (2014)
20. Yang, B., Yih, W., He, X., Gao, J., Deng, L.: Embedding Entities and Relations for Learning and Inference in Knowledge Bases. ICLR (Poster) (2015)
21. Yu, X., et al.: Recommendation in heterogeneous information networks with implicit user feedback. RecSys 2013, pp. 347–350 (2013)
22. Zhang, F., Yuan, N., Lian, D., Xie, X., Ma, W.: Collaborative Knowledge Base Embedding for Recommender Systems. KDD 2016, pp. 353–362 (2016)
23. Zhang, Y., Ai, Q., Chen, X., Wang, P.: Learning over Knowledge-Base Embeddings for Recommendation. CoRR abs/1803.06540 (2018)
24. Liu, C.H.B., Chamberlain, B.P., McCoy, E.J.: What is the value of experimentation and measurement? Data Sci. Eng. **5**(2), 152–167 (2020). https://doi.org/10.1007/s41019-020-00121-5

25. Luo, J., Xiao, S., Jiang, S., Gao, H., Xiao, Y.: ripple2vec: node embedding with ripple distance of structures. Data Sci. Eng. **7**(2), 156–174 (2022)
26. Wu, S., Zhang, Y., Gao, C., Bian, K., Cui, B.: GARG: anonymous recommendation of point-of-interest in mobile networks by graph convolution network. Data Sci. Eng. **5**(4), 433–447 (2020). https://doi.org/10.1007/s41019-020-00135-z
27. Liu, Y., Li, B., Zang, Y. et al.: A Knowledge-Aware Recommender with Attention-Enhanced Dynamic Convolutional Network. CIKM 2021, pp. 1079–1088 (2021)
28. Sikos, L.F., Philp, D.: Provenance-aware knowledge representation: a survey of data models and contextualized knowledge graphs. Data Sci. Eng. **5**(3), 293–316 (2020). https://doi.org/10.1007/s41019-020-00118-0