World Wide Web (2021) 24:1769–1789
https://doi.org/10.1007/s11280-021-00912-4

Path-enhanced explainable recommendation with knowledge graphs

Yafan Huang[1] · Feng Zhao[1] · Xiangyu Gui[1] · Hai Jin[1]

Received: 26 October 2020 / Revised: 20 February 2021 / Accepted: 10 June 2021 /
Published online: 2 July 2021

Abstract

Recommender systems, which are used to predict user requirements precisely, play a vital role in the modern internet industry. As an effective tool with rich semantics, knowledge graphs have recently attracted growing research attention in enhancing recommendation results. By mining multihop relations (*i.e.*, paths) between user-item interactions within a knowledge graph, implicit user preferences and other side information can be clearly revealed. Nevertheless, existing knowledge graph-based recommendation methods have two fundamental limitations. First, the indiscriminate utilization of user-item path sets conveys unclear information and negatively influences explainability. Moreover, obtaining reliable recommendation results with these methods requires large amounts of prior knowledge, which indicates that they show poor performance in terms of accuracy and handling cold-start issues. To address these issues, we propose a novel model called the *Path-enhanced Recurrent Network* (PeRN). Specifically, PeRN integrates a recurrent neural network encoder with a metapath-based entropy encoder to increase explainability and accuracy and reduce cold-start costs. The recurrent network encoder has a strong ability to represent sequential path semantics in a knowledge graph, while the entropy encoder, as an efficient statistical analysis tool, leverages metapath information to differentiate paths in a single user-item interaction. A path extraction algorithm with a bidirectional scheme is also proposed to make PeRN more feasible. The experimental results on two real-world datasets demonstrate our significant improvements with reasonable explanations, promising accuracy and a minimal amount of prior knowledge compared with several state-of-the-art baselines.

Keywords Recommender system · Knowledge graph · Metapath · Recurrent neural network

This article belongs to the Topical Collection: *Special Issue on Explainability in the Web*
Guest Editors: Guandong Xu, Hongzhi Yin, Irwin King, and Lin Li

✉ Feng Zhao
zhaof@hust.edu.cn

[1] National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China

1 Introduction

Recommendation systems (RSs), which aim to determine user preferences and provide items that the user might be interested in, have undergone rapid growth during the past decade in various fields, such as search engines, video portals, and e-commerce [4, 7, 22]. As a significant recommendation approach, traditional embedding-based methods have achieved impressive results by embedding user-item data into a low-dimensional continuous vector space and harnessing abundant side information. Despite their developments for several popular benchmarks, the recommendation results are calculated between latent vectors and do not take their relations into account, which leads to the poor explainability of embedding-based methods.

To address this issue, knowledge graph (KG), which is a well-structured auxiliary data format that evolved from semantic webs, has naturally been applied in recommendation systems to boost their reasoning ability and explainability [19, 27]. Information in a KG is organized as triples (*head, relation, tail*), and the *head* and *tail* entities are combined with relations. In KG-enhanced recommendation methods, all the users and items are regarded as entities together with other background information. By exploiting multihop relations from a target user entity within a KG, a user preference and its semantics can be explicitly revealed. Such a correlation is recorded as a path. Clearly, each target user entity has copious paths that lead to different items that the user may like, offering precise textual information for biclassification and top-K recommendation tasks. As a result, these path-based methods [10, 21] quickly received considerably more research interest than traditional translation-based methods such as TransE [2], TransH [25] and the collaborative knowledge graph embedding method (CKE) [32].

In terms of evaluating a recommender system augmented by a KG, we consider it crucial that convincing recommendation results be accurate and explainable. However, none of the existing path-based methods can satisfy the above two conditions at the same time. First, they neglect the difference between paths in the same user-item interaction. Taking a simple case in the music recommendation field as an example, the act "$(user, likes, song) \wedge (song, sung\_by, singer) \wedge (singer, sing, song)$" obviously contributes more to results than the act "$(user, likes, song) \wedge (song, sung\_by, singer) \wedge (singer, is\_broth-er\_of, singer) \wedge (singer, sing, song)$" – the latter is uncommon and less credible in explaining why this $song$ should be recommended to this $user$. Indiscriminately processing the paths reduces information utilization and negatively affects explainability. Second, existing path-based methods are not as impressive in terms of accuracy and can be even worse than some traditional embedding-based models. Such defects, more seriously, increase cold-start costs in some top-k recommendation tasks. In a *knowledge-aware path recurrent network* (KPRN) [24], the prior knowledge that is used to complete the KG accounts for 50% of the original dataset, making it difficult to apply the model in industry (Figure 1).

To bridge these gaps, a new recommendation model, named the *Path-enhanced Recurrent Network* (PeRN), is proposed that extracts not only the path information in a KG but also a metapath set in order to differentiate the path contribution to one user-item pair. Inspired by previous work [33], we innovatively use a metapath, a general concept in a heterogeneous information network (HIN), to generalize the structure of complicated paths. The metapath is a path schema consisting of a series of entity-type data and relation data. By using a metapath, all the paths in a dataset can be abstracted as several kinds of user habits, which will be used to calculate credibility values in the entropy encoder in our model. Thus, the confidence ratio between different paths in the same user-item pair can be obtained. For

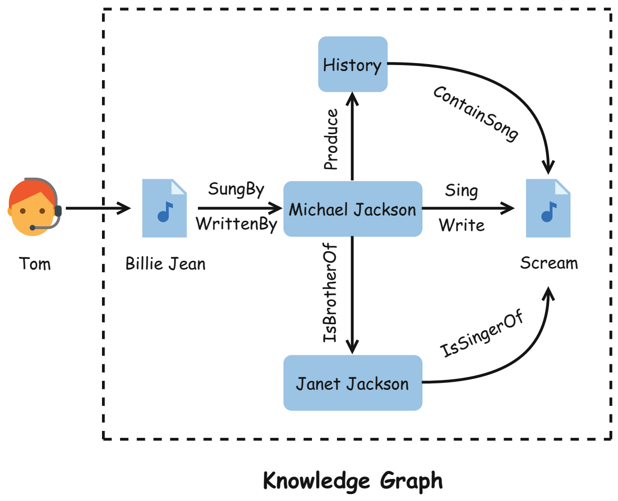


Figure 1 A toy example to illustrate why incorporating "relations" into recommendations can enhance explainability. In this music knowledge graph, there are different relations between entities. Finding all the relations related to "If Tom likes Scream" can bring more insights and make the result more reasonable

the path set, we adopt a bidirectional long short-term memory (bi-LSTM) network and a two-layer fully connected perceptron to compute sequential entity and relation vectors to determine a score. Afterwards, a weighted pooling layer is constructed to combine these path scores and credibility values from the entropy encoder to obtain a predicted result. To learn the parameters effectively, we propose using a logarithmic loss function along with a ridge penalty, i.e., L_2 regularization, to train our model. Furthermore, we design a bidirectional search method, which is also metapath-aided, to enhance the efficiency of path extraction and make PeRN easier to use.

To validate the ability of PeRN in recommendation accuracy and solve the cold-start issue, we conduct extensive experiments on two real-world datasets. Additionally, we give a user-item interaction example that is randomly chosen from the dataset to illustrate the increased explainability of our model. The main technical contributions of this paper include the following:

- An end-to-end model, PeRN, with an entropy encoder is proposed to enhance the explainability of the recommendations. The metapath, which helps to extract and encode user habits, is innovatively applied in general path-based methods. Our metapath-based path method avoids the limitation that the paths cannot be differentiated in one user-item interaction. Compared with traditional path-based models, the proposed PeRN achieves more expressive explainability and successfully reduces the cold-start costs.
- A novel path extraction method is proposed that is augmented with a bidirectional strategy. Similar to the Sentinel algorithm, the proposed bidirectional strategy can efficiently extract path data from a KG, which makes it possible to apply PeRN in real-world scenarios. Based on this, we also perform extensive experiments on two real-world datasets from biclassification and top-K perspectives. Our proposed PeRN outperforms several representative baselines in terms of accuracy. These demonstra-

tions also highlight the importance of integrating KGs into recommendations and verify the practicality of our proposed methods.

The rest of this paper is arranged as follows: Section 2 gives a brief review of two kinds of related works. Then, Section 3 describes the framework and notation of the PeRN model, and Section 4 explains PeRN in detail. The experimental results that verify this model are shown in Section 5. Finally, Section 6 contains conclusions and some ideas for future work.

2 Related works

This section provides a general summary of several state-of-the-art methods of integrating KGs into recommendations, which can be largely divided into translation-based and path-based methods.

2.1 Translation-based methods

Prior research has proposed various techniques [1, 2, 5, 13, 15, 25] for embedding a KG into a low-dimensional vector space in a way that makes the KG computable. These methods can be roughly divided into two categories [23]: 1) translational distance models, including translation-based and other distance models, and 2) semantic matching models, including tensor-factorization-based and neural-network-based models. Translation-based models such as TransE [2] and its variants [12, 13, 25], which use the idea of translation to transform entities and relations into vectors to embed KGs, have been widely used in KG recommendation because of their simplicity and effectiveness. CKE [32] first employed Bayesian TransR [13] to generate the user latent vector and KG-aided item latent vector to collaboratively learn the predicted result. DKN [27], leveraging four different translation-based models [2, 12, 13, 25] to embed a KG and enrich the side information, also applied an attention-based deep convolutional neural network to the news recommendation field. More recently, the translation-based user preference model (TUP) [3] employed TransH [25] to predict user preferences from a KG to improve recommender systems.

Such translation-based methods significantly increase the accuracy of the results. However, they fail to explore the correlations (i.e., multi-hop relations) among user-item pairs. In other words, these methods fail to explain why the user has interest in these items. Thus, we argue that these methods lack explainability and reasoning ability in recommendations.

2.2 Path-based methods

Regarding path-based methods, Yu et al. [31] integrated a heterogeneous information network into matrix factorization for personalized entity recommendation. This was a new idea that not only first introduced user-item-path conception to recommender systems but also inspired other researchers to apply paths in KGs for convincing recommendations. As a consequence, various path-based approaches [11, 24, 33] have been developed to memorize sequential implicit information in KGs. Knowledge-based sequential recommendation (KSR) [11] incorporates a gated recurrent unit (GRU) and key-value memory network (KV-MN) to capture sequential user preferences from a knowledge base, while recurrent knowledge graph embedding (RKGE) [19] leverages recurrent network batches to embed path semantics to increase interpretability. In KPRN [24], a long short-term memory (LSTM) network is utilized to encode paths and explore the connectivity between users and items. Additionally, explainable interaction-driven user modeling (EIUM) [10] extends this idea to a multimodal knowledge base and designs a self-attention matrix to mine deep information from a path. With the development of graph neural networks (GNN), there are also some path-based methods adopting GNN as their path encoder. Path conditioned Graph Convolutional Network (PGCN) [30] uses graph convolutional network (GCN) to encode path information between entities, while Price-aware User Preference-modeling (PUP) [34] adopt GCN as a part of their self-defined encoder-decoder to record pairwise interactions for better recommendation.

Although these methods improve the explainability and reasoning ability of recommender systems, there is still a severe defect: the underutilization of mining semantics in paths. This flaw also causes poor performance in dealing with cold-start issues: the prior knowledge needed for completing a KG is extremely costly. Moreover, path extraction is also a time-consuming and labor-intensive step in path-based methods, as the time complexity of the algorithm grows exponentially with the length of the path.

3 Model design

3.1 Framework of PeRN

With a given knowledge graph and a target user-item interaction, the path set and its metapath set can be extracted by a bidirectional path extraction algorithm. After an embedding step, these paths and metapaths are processed to obtain a certain score and a weight by a recurrent network encoder and an entropy encoder, respectively. Combining these scores by using a weighted pooling layer, the output of PeRN, a recommendation prediction, is obtained. The overall framework of PeRN is illustrated in Figure 2.

PeRN contains four key components: a *bidirectional path extraction* algorithm to extract paths effectively from a KG between two target entities (*i.e.*, a user and item), a *recurrent network encoder* that is based on a Bi-LSTM neural network and a fully connected layer to encode the path information as a certain value, an *entropy encoder* that adopts information gain and metapaths to assign a weighting score to each path in one user-item interaction, and a *weighted pooling* layer that combines the above values and weighting score to obtain a predicted result.

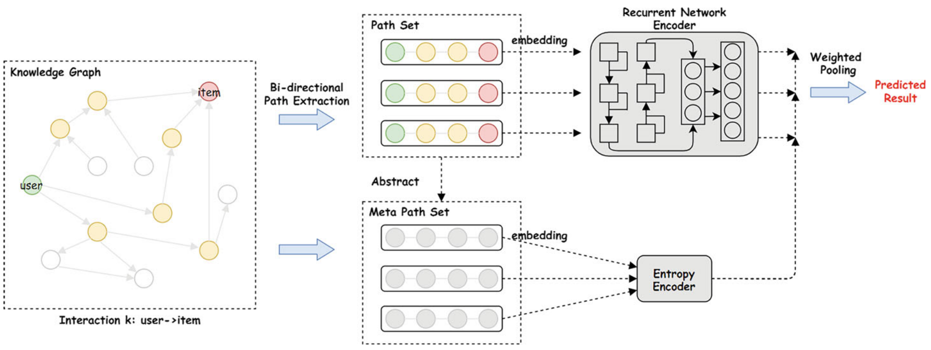


Figure 2 Framework of PeRN

3.2 Problem definition

Before describing our model, we formally define the notation used throughout this paper in Table 1. Similar to other recommender systems, we let $\mathcal{U} = \{u_1, u_2, ..., u_{|\mathcal{U}|}\}$ and $\mathcal{I} = \{i_1, i_2, ..., i_{|\mathcal{I}|}\}$ denote the user set and item set, respectively. $\mathcal{A} = \{(u, i)|u \in \mathcal{U}, i \in \mathcal{I}\} = \{a_1, a_2, ..., a_{|\mathcal{A}|}\}$ represents all the interactions between users and items in our dataset.

Definition 1 Knowledge Graph. As the entity set $\mathcal{E}$ and relation set $\mathcal{R}$ have been defined, a knowledge graph (KG) can be defined as $\mathcal{KG} = \{(h, r, t)|h, t \in \mathcal{E}, r \in \mathcal{R}\}$, where (h, r, t) is a triple combining a head entity h and tail entity t through a relation r. Here, every user and item in $\mathcal{U}$ and $\mathcal{I}$ can be searched as an entity in $\mathcal{KG}$, which makes the extraction of paths between user-item interactions possible. By abstracting entities in $\mathcal{E}$ to entity types (shown as the function typeof() in the table above), the schema of $\mathcal{KG}$ can be revealed along with relations, and this is expressed as a two-dimensional matrix $\mathcal{G}$.

Definition 2 Path and **Metapath.** Given an interaction $a_k = (u_m, i_n)$, a sequence of triples that connect user u_m and item i_n can be found in $\mathcal{KG}$ as $\{(u_m, r_1, e_1), (e_1, r_2, e_2), ..., (e_{l-1}, r_l, i_n)\}$, which we record as a path: $p = u_m \xrightarrow{r_1} e_1 \xrightarrow{r_2} e_2 ... \xrightarrow{r_l} i_n$. Therefore, all the qualified paths of an interaction a_k and interaction set $\mathcal{A}$ are denoted as $P_k = \{p_1, p_2, ..., p_{|P_k|}\}$ and $\mathcal{P} = \{P_1, P_2, ...P_{|\mathcal{A}|}\}$, respectively. As each path forms a metapath by abstracting its entities to entity types, we denote mp as the metapath of path p and $MP_k = \{mp_1, mp_2, ..., mp_{|MP_k|}\}$ as the metapath set of

Table 1 Notation

Description	Notation
User set	$\mathcal{U} = \{u_1, u_2, ...u_{|\mathcal{U}|}\}$
Item set	$\mathcal{I} = \{i_1, i_2, ...i_{|\mathcal{I}|}\}$
User-item interaction set	$\mathcal{A} = \{a_1, a_2, ...a_{|\mathcal{A}|}\}$
Entity set	$\mathcal{E} = \{e_1, e_2, ...e_{|\mathcal{E}|}\}$
Relation set	$\mathcal{R} = \{r_1, r_2, ...r_{|\mathcal{R}|}\}$
Knowledge graph	$\mathcal{KG} = \{(h, r, t)|h, t \in \mathcal{E}, r \in \mathcal{R}\}$
Schema graph	$\mathcal{G} = \mathbb{R}^{g \times g}, g = |\text{typeof}(\mathcal{E})|$
Interaction a_k in KG	$a_k = (u_m, i_n), u_m \in \mathcal{E}, i_n \in \mathcal{R}$
A path p between a_k	$p = u_m \xrightarrow{r_1} e_1 \xrightarrow{r_2} e_2, ... \xrightarrow{r_l} i_n$
Metapath mp between a_k	$mp = \text{typeof}(u_m) \xrightarrow{r_1}, ...\text{typeof}(i_n)$
Paths between a_k	$P_k = \{p_1, p_2, ..., p_{|P_k|}\}$
Metapaths between a_k	$MP_k = \{mp_1, mp_2, ...mp_{|MP_k|}\}$
Path set	$\mathcal{P} = \{P_1, P_2, ...P_{|\mathcal{A}|}\}$
Metapath set	$\mathcal{M} = \{mp_1, mp_2, ..., mp_{|\mathcal{M}|}\}$
Hidden state vectors	$\overrightarrow{h}_{l+1}, \overleftarrow{h}_{l+1}, h$
Weight matrix	$W_f, W_i, W_C, W_o, W_1, W_2$
Sigmoid function	σ
Loss function	$\mathcal{L}$

path set P_k (and of interaction a_k). The metapath of all interactions, denoted as a set $\mathcal{M}$, can be obtained by a certain traversal of the schema graph $\mathcal{G}$.

In addition, there are three points to be emphasized: 1) $\mathcal{U}, \mathcal{I} \subsetneqq \mathcal{E}$ and $\mathcal{U} \cap \mathcal{I} = \varnothing$. 2) MP_k and P_k are both generated from a_k, so $|MP_k| \leqslant |P_k|$, and equality holds when all path types in $|P_k|$ are different. 3) $\mathcal{M} = MP_1 \cup MP_2 \cup ... \cup MP_{|\mathcal{A}|}$.

Definition 3 PeRN Task. With the given user-item interaction a_k and its path set P_k, the goal of PeRN is formulated as follows:

$$\hat{y}_k = f_{\Delta}(P_k), \tag{1}$$

where $\hat{y}_k$ is the predicted score of interaction a_k and f denotes the function of PeRN with parameters Δ.

4 Path-enhanced recurrent network

In this section, we thoroughly describe and elaborate our proposed PeRN for incorporating KGs into recommendations. First, we design a bidirectional path extraction algorithm to boost the efficiency of discovering path sets between user and item entities in a KG. Then, we adopt a Bi-LSTM network as a model to embed the path set and remember it as a set of predicted scores. Furthermore, the entropy encoder is created to differentiate the contributions of the paths in a path set by analyzing the information gain of their metapath. Finally, a weighted pooling layer and optimization steps are used to combine the scores and learning.

4.1 Bidirectional path extraction

A KG generally contains millions of entities and relations, which indicates that it is labor-intensive and time-consuming to find all paths between two entities. In addition, the difficulty of searching a path increases exponentially with its length, which makes path extraction even more difficult.

To address this issue, a metapath-aided bidirectional path extraction algorithm is proposed to retrieve all qualified paths, as described in Algorithm 1. As previous work [18] discussed, we regard paths that are longer than six hops as noise. Thus, this bidirectional scheme can change the complexity of the search step from a maximum of six hops to a maximum of three hops. In the preliminary steps, we first abstract the KG to its schema graph by changing the entities in the triples to entity types. By removing all duplicates among these processed triples, the schema graph can be displayed via a matrix in which every dimension has a list of all entity types. The elements in the matrix can store the types of relations between the entity types. Accordingly, the whole metapath set $\mathcal{M}$ can be retrieved by traversing this directed graph. Given a knowledge graph $\mathcal{KG}$, user-item interaction set $\mathcal{A}$ and metapath set $\mathcal{M}$, Algorithm 1 can help us find a whole path set $\mathcal{P}$. After the initialization of $\mathcal{P}$, for each interaction a_k, we adopt the depth-first-search (DFS) idea to retrieve candidate paths and the metapath-aided idea to choose target paths bidirectionally. In fact, candidate sets S_1 and S_2 can be used in parallel for different metapaths so that we significantly enhance the efficiency of path extraction by designing a multithreaded program.

Algorithm 1 Bidirectional path extraction algorithm.

```
Input: Knowledge graph $\mathcal{KG} = \{(h, r, t)|h, t \in \mathcal{E}, r \in \mathcal{R}\}$, user-item interaction set
       $\mathcal{A} = \{a_1, a_2, ...a_{|\mathcal{A}|}\}$,  metapath set $\mathcal{M} = \{mp_1, mp_2, ..., mp_{|\mathcal{M}|}\}$.
Output: Path set $\mathcal{P} = \{P_1, P_2, ..., P_{|\mathcal{A}|}\}$.
1  Initialize: $\mathcal{P} \leftarrow \varnothing$ ;
2  for each $a_k = (u_m, i_n)$ in $\mathcal{A}$ do
3      $P_k \leftarrow \varnothing$;
4      $S_1 \leftarrow$ retrieve all paths by head = $u_m$ within 3 hops;
5      $S_2 \leftarrow$ retrieve all paths by head = $i_n$ within 3 hops;
6      for each metapath $mp$ in $\mathcal{M}$ do
7          $l \leftarrow$ length of $mp$;
8          $P_1 \leftarrow \varnothing$ // left sub-path set;
9          $P_2 \leftarrow \varnothing$ // right sub-path set;
10         for each $p_l$ in $S_1$ do
11             if $p_l$ satisfies $mp[0 \rightarrow 2 * \lfloor l/2 \rfloor]$ then
12                 Add $p_l$ to $P_1$;
13         for each $p_r$ in $S_2$ do
14             if $p_r$ satisfies $mp[2 * \lfloor l/2 \rfloor \rightarrow 2 * l]$ then
15                 Add $p_r$ to $P_2$;
16         for each $p_l$, $p_r$ in $P_1$, $P_2$ do
17             if $p_l[tail] = p_r[tail]$ then
18                 $p \leftarrow$ combine $p_l$ and reverse($p_r$);
19                 Add $p$ to $P_k$;
20     Add $P_k$ to $\mathcal{P}$;
21 return $\mathcal{P}$
```

4.2 Recurrent network encoder

With the increasing development of deep learning models, recurrent neural networks (RNNs) have become increasingly widely used in processing sequence data such as path information [19, 24]. In the PeRN model, as relations in paths are not always in one direction – for example (Michael Jackson, Cooperate_with, Janet Jackson) and (Janet Jackson, Cooperate_with, Michael Jackson) – we choose a bi-LSTM-based model, as illustrated in Figure 3, to better sequence information and output the predicted score of the path. In other words, we input a path p and output its predicted score s from this network.

A path here can be recorded as a sequence of entities and relations such as $p = [u_m, r_1, e_1, r_2, e_2..., r_l, i_n]$ as well as a number of relations less than or equal to six. To embed this multihop data into a series of vectors, we first transform this l-hop path to $l + 1$ consecutive vectors such as $(u_m, \text{typeof}(u_m), r_1), (e_1, \text{typeof}(e_1), r_2), ...(e_{l-1}, \text{typeof}(e_{l-1}), r_l), (i_n, \text{typeof}(i_n), null)$. Then, the entity index, a number from 0 to millions, is mapped to a $d - 2$-dimensional vector with a similar number. After concatenating the entity vector with the entity type index and relation index, we can obtain a d-dimensional vector representing a hop on the

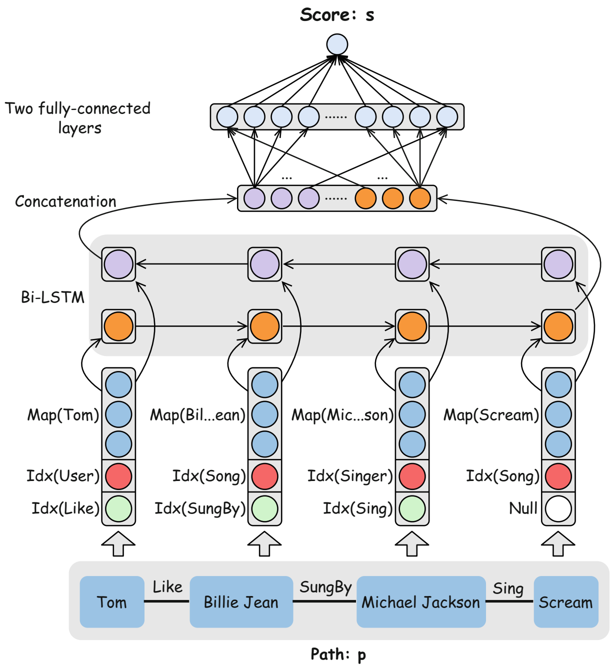


Figure 3 The architecture of the recurrent network encoder in PeRN

path. The q-th vector in the path can be defined via the following equation:

$$\alpha_q = \mathrm{Map}(e_{q-1}) \oplus e'_{q-1} \oplus r_q, \tag{2}$$

where $\oplus$ is the concatenation operation and e' is the type of entity e. After this step, path p is represented as a vector set $\{\alpha_1, \alpha_2, ..., \alpha_{l+1}\}$, which can be taken as the input of this model. With a strong ability to remember sequential semantics [14], LSTM [8] is chosen to be the main body of the recurrent network encoder, and the q-th vector α_q of the input vector set is computed as follows:

$$\begin{aligned} f_q &= \sigma(W_f \dot{(} h_{q-1} \oplus \alpha_q) + b_f) \\ i_q &= \sigma(W_i \dot{(} h_{q-1} \oplus \alpha_q) + b_i) \\ \widetilde{C}_q &= \tanh(W_C \dot{(} h_{q-1} \oplus \alpha_q) + b_C) \\ C_q &= C_{q-1} \odot f_q + \widetilde{C}_q \odot i_q \\ o_q &= \sigma(W_o \dot{(} h_{q-1} \oplus \alpha_q) + b_o) \\ h_q &= o_q \odot \tanh(C_q) \end{aligned} \tag{3}$$

where f_q, i_q and $o_q \in \mathbb{R}^{d'}$ denote the forget, input and output gates; $\widetilde{C}_q$ and $C_q \in \mathbb{R}^{d'}$ denote the candidate value vector and memory state vector; W_f, W_i, W_C and $W_o \in \mathbb{R}^{d' \times (d'+d)}$ are weight matrices initialized with random values; b_f, b_i, b_C and b_o are the bias vectors of each gate or cell; σ and tanh denote sigmoid and hyperbolic tangent activation functions; and $\odot$ and $\oplus$ represent the Hadamard product and concatenation, respectively. Consequently, the hidden state vector $h_q \in \mathbb{R}^{d'}$ of the q-th step is obtained by the last hidden state h_{q-1} given α_q. After $l+1$ iterations, the last hidden state vector h_{l+1} can be considered to hold the sequential path information. To explain more precisely, we simplify this process as the following equation:

$$h_{l+1} = \text{LSTM}([\alpha_1, \alpha_2, ..., \alpha_{l+1}]). \tag{4}$$

As shown in Figure 2, our PeRN model takes bi-LSTM into consideration; it adopts a forward LSTM and a backward LSTM and then concatenates the bidirectional results to remember path information more reliably. This step can be formulated as the equations below:

$$\begin{aligned} \overrightarrow{h}_{l+1} &= \text{LSTM}([\alpha_1, \alpha_2, ..., \alpha_{l+1}]) \\ \overleftarrow{h}_{l+1} &= \text{LSTM}([\alpha_{l+1}, ..., \alpha_2, \alpha_1]) \\ h &= \overrightarrow{h}_{l+1} \oplus \overleftarrow{h}_{l+1}. \end{aligned} \tag{5}$$

After embedding path p into a representative vector h, the final step of the recurrent network encoder is to convert it to a predicted score by establishing a simple neural network with two fully connected layers. Therefore, the score s of the path can be calculated by the following equation:

$$s = W_2^{\mathrm{T}} \text{ReLU}\left(W_1^{\mathrm{T}} h\right). \tag{6}$$

Here, W_1^{T} and W_2^{T} represent the coefficient weights of layer 1 and layer 2, respectively, and we adopt a rectified linear unit (ReLU) as the activation function in each neuron with omitted bias. In fact, the score s is a 1×1 matrix, and we directly treat it as a scalar for simplicity.

4.3 Entropy encoder

Given an interaction $a_k = (u_m, i_n)$ and its extracted path set $P_k = \{p_1, p_2, ..., p_{|P_k|}\}$, the scores of each path can be stored in a set $S_k = \{s_1, s_2, ..., s_{|P_k|}\}$. By abstracting the entity instances to entity types in a path, the metapath set of a_k can be denoted as $MP_k = \{mp_1, mp_2, ..., mp_{|MP_k|}\}$, where $|MP_k| \leqslant |P_k|$ (*cf.* Section 3). Clearly, it does not make sense to predict a_k by taking the weighted averages of the path scores (*cf.* Section 1). To address this issue and increase the explainability, we design an information entropy-based method.

Inspired by recent applications and advances in the computer vision field [17, 20], we design an entropy-based weighting encoder to differentiate path contributions by computing the information gain for specific interactions. In practice, all the interactions are either positive feedback or negative feedback, which indicates that each path p in a path set P_k shares the same target, 1 or 0. Here, we collectively define a path or metapath with target 1 as a confidence path (CP) and otherwise as a non-confidence path (NP). By traversing and counting all paths p in the whole path set $\mathcal{P}$, it is possible to obtain the frequency of "p is a CP" considering that it is a binary classification problem. Therefore, the information entropy of the event "p is a CP" (denoted as event D, which takes the values 0 and 1) is defined as follows:

$$\text{Ent(D)} = -\sum_{j \in 0,1} \text{P(D} = j) \log_2 \text{P(D} = j). \tag{7}$$

Although there are millions of paths in $\mathcal{P}$, the number of metapaths is limited, and we denote it as an integer m. Therefore, we can endow path m with Boolean features to determine the type of its metapath and further judge its contribution to event D. Similar to (5), for feature $E_g (g \in [1, m])$, we can find its conditional entropy for event D as follows:

$$\mathrm{Ent(D}|E_g) = \sum_{i \in 0,1} \mathrm{P}(E_g{=}i)\mathrm{Ent(D}|E_g{=}i), \tag{8}$$

where $\mathrm{Ent(D}|E_g = i)$ is the conditional entropy; it is obtained by fixing $E_g{=}i$ and is written as the equation below:

$$\mathrm{Ent(D}|E_g = i) = -\sum_{j \in 0,1} \mathrm{P(D} = j|E_g = i) \log_2 \mathrm{P(D} = j|E_g = i). \tag{9}$$

In this way, the information gain of event D for a given feature E_g can be obtained:

$$\mathrm{Gain(D}, E_g) = \mathrm{Ent(D)} - \mathrm{Ent(D}|E_g). \tag{10}$$

For a given interaction a_k and its metapath set MP_k, each $mp_i (i \in [1, |MP_k|])$ can determine its unique corresponding information gain by matching $E_g{=}mp_i$. Here, we normalize the information gain of each metapath to obtain its weight:

$$w_i = \frac{\mathrm{Gain(D,}E_g = mp_i)}{\Sigma_{j=1}^{|MP_k|}\mathrm{Gain(D,}E_g = mp_j)}, \tag{11}$$

where w_i is the weight of the i-th path p_i in the path set of a_k; we can further differentiate the score s_i in S_k. All the weights of the same interaction a_k are put in a weight set W_k for a later step.

4.4 Objective of optimization

After obtaining a score set $S_k = \{s_1, s_2, ..., s_{|P_k|}\}$ and its corresponding weight set $W_k = \{w_1, w_2, ..., w_{|P_k|}\}$, we adopt a weighted pooling layer that combines them to obtain the final predictive score, which is formulated as:

$$\hat{y}_k = \sigma\left(\sum_{i=1}^{|P_k|} w_i s_i\right), \tag{12}$$

where σ is a sigmoid activation function that maps the final score to a range of 0 to 1. The weighted pooling layer, which can be regarded as an attention mechanism, combines the path scores with their own weights, indicating the importance of the paths in the target user-item interactions.

Since all the interactions in $\mathcal{A}$ can be considered positive feedback or negative feedback (*cf.* Section 4.3), we regard our recommendation task as a binary classification task with 0 representing negative and 1 positive, similar to previous work [24]. We adopt cross-entropy loss to optimize our result. With the given interaction a, our loss function is defined as follows:

$$\mathcal{L} = -\sum_{a \in \mathcal{A}} (y \log \hat{y} + (1 - y) \log(1 - \hat{y})), \tag{13}$$

where $\mathcal{A} = \{a_1, a_2, ..., a_{|\mathcal{A}|}\}$ is the whole user-item interaction set in the dataset and y and $\hat{y}$ represent the observed user-item feedback and the predictive score of a, respectively. We also conduct L_2 regularization (i.e., the ridge penalty) on the parameters of our PeRN, which is omitted here for simplicity.

5 Experiments

We perform various experiments on 2 real-world recommendation datasets, which are based on music recommendation and movie recommendation scenarios, to evaluate our PeRN with several state-of-the-art baselines. To more realistically construct a KG for recommendation and mine user preferences by path, we adopt two benchmark datasets, and the statistics are shown in Table 2.

- KKBox [1], which is a music domain recommendation dataset from the WSDM Cup Challenge 2018 and is provided by the KKBox Music Streaming Service.
- IM-1M, composed of Internet Movie Database (IMDb)[2] and MovieLens-1M [3] datasets, is a common movie recommendation dataset that has recently been widely used in KG-enhanced recommendation tasks [19, 24].

To evaluate our PeRN more comprehensively and demonstrate its rationality, we use evaluation metrics from the perspectives of the binary classification recommendation task performance, the top-K recommendation task performance and the ability to handle cold-start issues, which are shown below:

- Precision (**P**), recall (**R**), F_1**-score**, and area under the curve (**AUC**): These five metrics are adopted to represent the overall accuracy of the biclassification task. Precision and recall are commonly used to address the imbalance of positive and negative samples. The F_1-score is the harmonic mean of precision and recall. The AUC, the threshold of which we set in the range of 0.1 to 0.9, is calculated as in the following equation:

$$\mathrm{AUC} = \frac{\sum_{i \in positive}(P + N + 1 - rank_i) - \frac{P(1+P)}{2}}{P \times N}, \tag{14}$$

  where P and N are the numbers of positive and negative interactions, respectively, and $rank_i$ denotes the rank of i's predicted score.
- Normalized discounted cumulative gain (**NDCG@K**): As the most common metric in the top-k recommendation task, the NDCG@K evaluates the model performance in terms of position influence, which is calculated as follows:

$$\mathrm{NDCG@K} = \frac{1}{\alpha}\mathrm{DCG@K} = \frac{1}{\alpha}\sum_{i=1}^{K}\frac{2^{rel_i} - 1}{\log_2(i + 1)}, \tag{15}$$

  where rel_i indicates the relevance of the recommendation at position i, K is the size of the target list, and α is a standardization constant that is the maximum value of the DCG@K. Considering that the sizes of these two datasets are different, we set K in KKBox and IM-1M as {3, 5, 10, 15} and {2, 6, 10, 12}, respectively.
- Percentage of interactions used to complete the KG (**PcKG**): In measuring the cold-start cost, we set the PcKG in the range {10%, 20%, 30%, 40%, 50%}. Here, a lower PcKG with a higher score of the other metrics indicates a better ability to handle cold-start issues.

Table 2 Statistics of KKBox and IM-1M

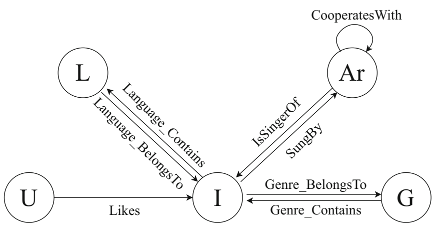
Dataset		KKBox	IM-1M
User-Item Interaction	# of Users	34,403	6,040
	# of Items	2,296,320	3,274
	# of Interactions	3,696,465	370,023
	Data Density	0.0047%	1.87%
Knowledge Graph	# of Entities	2,562,937	15,439
	# of Entity Types	5	5
	# of Relation Types	8	9
	# of Triples	16,237,068	442,409
Path	# of Paths	41,400,408	345,344
	Avg. Path Length	5.11	4.74
	# of Metapath Types	21	46
	Avg. Metapath Length	5	5.37

5.1 Experimental setup

The fundamental part of obtaining the KG-enhanced recommendation dataset is constructing the domain of the KG. First, the basic part of the KG can be selectively generated from the background information in the original dataset. Then, to integrate the user set into the KG, which makes finding paths between the users and items possible, a group of interactions should be added to the KG as compensation. KKBox not only provides a large amount of user-item interaction data with positive feedback 1 and negative feedback 0 but also several pieces of side information, such as the artist, lyricist, composer, and genre, which indicates that this music domain KG can be straightforwardly constructed. In IM-1M, the IMDb contains comprehensive auxiliary information such as the core contributors, duration, genre and budget of the movie, while MovieLens-1M provides more than 1,000,000 user-item interactions with rating scores {1, 2, 3, 4, 5}. Thus, a movie domain KG can be constructed by mapping the movie titles in IMDb and MovieLens-1M.

To better fit our proposed model, all the interactions in which the rating scores are 3 are omitted here to enable biclassification, and the other 4 scores are normalized for the top-K task. In the processed KKBox and IM-1M data, approximately 50% of the original interactions are treated as valid interactions (the exact numbers are shown in Table 2). The other part, which is used for completing the KG, can be used to measure the severity of the cold-start issue by the **P**ercentage of interactions used to **c**omplete the **KG** (**PcKG**). In terms of path data, we first provide our definition of the KG: the schema graphs of the music domain KG and movie domain KG that we designed are shown in Figure 4. In KKBox, U: user, I: item (song), L: language, Ar: artist, G: genre. In IM-1M, U: user, I: item (movie), Ac: actor, D: director, G: genre. In KKBox, we select the user, item (song), language, artist and genre as the target entity types. The relation “CooperatesWith” is extracted from the

[1] https://www.kaggle.com/c/kkbox-music-recommendation-challenge
[2] https://www.kaggle.com/suchitgupta60/imdb-data
[3] https://grouplens.org/datasets/movielens/



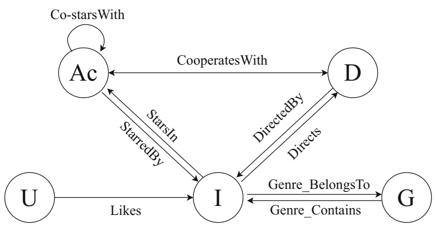


Figure 4 Schema graphs of KKBox (left) and IM-1M (right)

condition "multiple artists in one song". In IM-1M, we choose the user, item (movie), director, actor and genre as the target entity types. Additionally, there are "actor_1_name" and "actor_2_name" attributes of a movie, and we regard the relation between them as "Co-starsWith". Metapaths can be extracted by certain traversals in the schema graph, and paths are extracted by Algorithm 1.

Furthermore, we omit pretrained parameters in our proposed model to enable pair comparison with several baselines. To reasonably embed the paths into a vector space, we map the large fluctuating entity index to a 62-dimensional vector and concatenate it with the entity type index and relation type index. Therefore, the size of each input vector of the LSTM is 64, and each has a length of up to 7 (i.e., a 6-hop path). If the length of a path is less than 7, then it is completed with a zero vector. The batch size is 256 here. We adopt stochastic gradient descent (SGD) in the optimization step with a learning rate in the range $\{0.001, 0.01, 0.1, 0.5\}$, while the L_2 regularization coefficients are tuned in the range $\{10^{-5}$, 10^{-4}, 10^{-3}, $10^{-2}\}$. In our experiments, we compared our PeRN with the widely used recommendation methods below, which are based on matrix factorization (MF), factorization machines (FMs), KG translation, metapaths in a heterogeneous information network (HIN) and paths in a KG. The brief descriptions of these methods are as follows:

- MF [16]: This is a standard matrix factorization method with Bayesian personalized ranking (BPR) loss that regards interactions as elements in the user-item matrix to predict unknown interactions.
- AFM [28]: This method is a factorization model combined with an attention mechanism that excavates potential relations and interactions among user preferences and item features.
- RippleNet [26]: The key of this method is preference propagation, which regards the historical interests of users as a seed set and propagates user preferences in the KG that contain extra information.
- MEIRec [6]: This is a metapath-guided method that contains rich user-item information and interactions based on a HIN and utilizes structural information fully for intent recommendation.
- KPRN [24]: KPRN is a representative method for integrating paths into KG recommendations through recurrent neural networks. Although KPRN achieves great performance in both binary classification and top-K recommendation tasks, it has the severe cold-start issue that its PcKG can reach **50%**.

5.2 Performance evaluation

Table 3 and Figure 5 report our experimental results on binary classification recommendation and the ability to solve the cold-start issue in the top-K recommendation task,

Table 3 Summary of the performance on the binary classification recommendation task for all baselines and our proposed PeRN on the KKBox and IM-1M datasets

Metrics	P	R	F_1	AUC
Dataset	KKBox			
MF	0.509	0.528	0.518	0.511
AFM	0.517	0.533	0.525	0.536
RippleNet	0.699	0.732	0.715	0.762
MEIRec	0.753	0.774	0.763	0.819
KPRN	0.805	0.822	0.813	0.834
PeRN	**0.842**	**0.861**	**0.851**	**0.866**
Dataset	IM-1M			
MF	0.612	0.608	0.610	0.586
AFM	0.647	0.632	0.639	0.601
RippleNet	0.742	0.713	0.727	0.694
MEIRec	0.792	0.804	0.798	0.734
KPRN	**0.843**	0.826	0.834	0.812
PeRN	0.835	**0.871**	**0.853**	**0.851**

respectively. In Table 3, the bold numbers indicate the best result of each column. In Figure 5, '*' indicates that the PcKG of KPRN is consistently 50%.

For the biclassification task, regarding the binary classification recommendation issue, the main challenge is to achieve good performance on two user-item interaction matrices. In KKBox, the data density is only **0.0047%**, so it is extremely sparse and renders various traditional recommendation methods unusable. MF and AFM are recommendation methods based on matrix operations and have obtained surprising results when processing highly sparse KKBox data. RippleNet achieves significant improvements over MF and AFM by integrating the KG into user-item interactions, but its predictive results are not as appreciable as those of the path-based MEIRec and KPRN considering the high sparseness when embedding a KG. Our proposed PeRN substantially outperforms the state-of-the-art methods and shows the best performance in several binary classification recommendation evaluation metrics. In IM-1M, as the data density is 1.87% here, PeRN and several baselines achieve slightly better results in terms of the P, R, and F_1 values. In addition, the original rating scores in IM-1M are displayed in the normalized range {1, 2, 4, 5} rather than 1 for positive and 0 for negative, so the actual classification effects are not as excellent as expected, which is shown by a lower AUC score than for KKBox. Nonetheless, our proposed PeRN still has more promising results than the other baselines.

For the top-K task and cold-start costs, as Figure 5 illustrates, our proposed PeRN also has a better performance in alleviating the cold-start issue in the top-K recommendation task than the state-of-the-art path-based method KPRN. Here, KPRN utilizes 50% of the given target user-item interactions in both the IM-1M and KKBox datasets in the process of constructing the KG, which causes a severe cold-start issue. To demonstrate the ability of PeRN to handle cold-start issues in the top-K task, we adopt the PcKG, which is shown as a percentage, to evaluate our model against the baseline. Specifically, we use PcKG = {10%, 20%, 30%, 40%, 50%} of the interactions of each user in the original dataset, which aims to ensure that all the users can be found as entities in the KG to complete the music domain

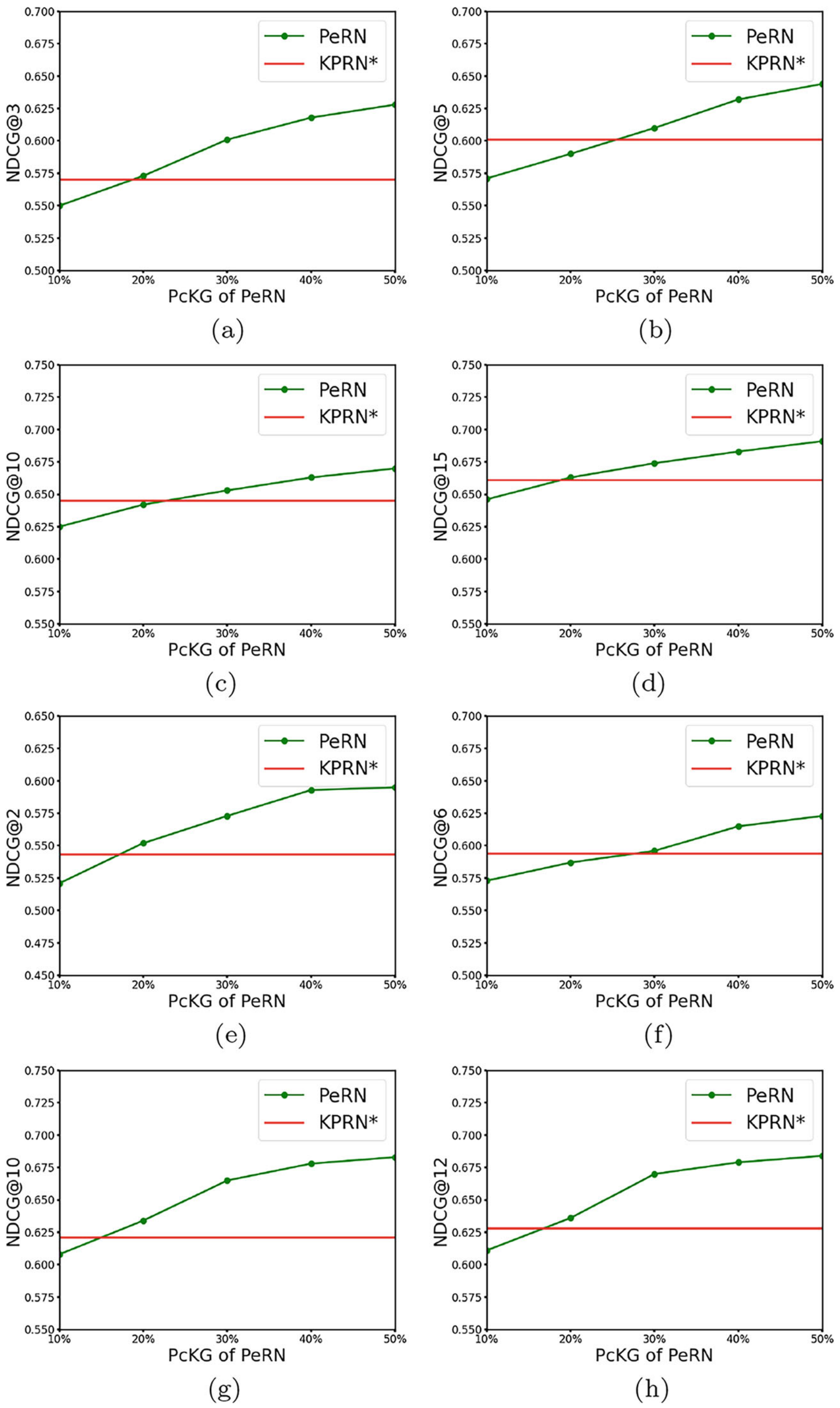


Figure 5 Performance of PeRN in the top-K task and cold-start costs, measured by NDCG@{3, 5, 10, 15} in KKBox and NDCG@{2, 6, 10, 12} in IM-1M

and movie domain KG. Every new completion has to re-extract all the paths to confirm the accuracy of the experiment. As measured by the NDCG@K – where K = {3, 5, 10, 15} in KKBox and K = {2, 6, 10, 12} in IM-1M considering that the size of the paths is different in each dataset – our PeRN is able to achieve a much more promising performance. In KKBox, with only a 20%-30% PcKG, it can reach a higher accuracy in the top-K recommendation task than KPRN with a 50% PcKG, while this number normally decreases to 10%-20% in IM-1M. In other words, we use only 30% of the prior knowledge on average and attain a better prediction result in the top-K recommendation task. Nevertheless, we must admit that these results should increase with a more thorough KG that remains to be designed and established. We hold a strong belief that reducing and refining prior knowledge is the key to decreasing cold-start costs.

5.3 Explainability analysis

In this section, we use a visualization example and a quantitative analysis to demonstrate PeRN's effectiveness in enhancing the explainability.

Firstly, we randomly select a user-item interaction from the IM-1M dataset and visualize its three paths, as shown in Figure 6, to illustrate the model's enhanced reasoning ability in a real scenario. The IDs of the user and target item are 3408 and 318, respectively. The red numbers denote the weighting score of each path. Some observations and analyses about "why user_3408 likes the movie The Shawshank Redemption" are presented below. As Figure 6 shows, three different paths between (U_id: 3408, I_id: 318) are clearly found,

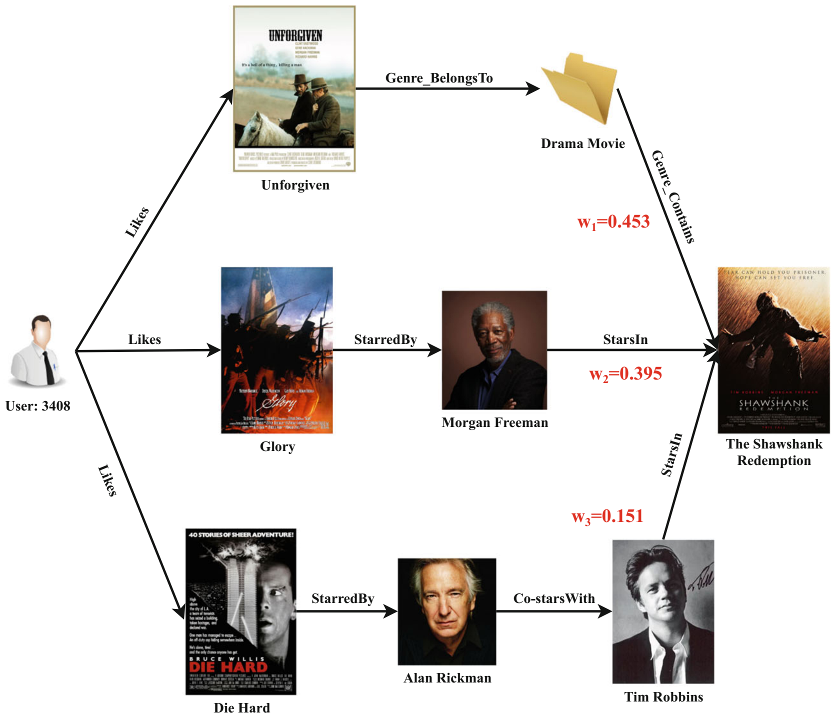


Figure 6 Visualization of the paths between selected user-item interactions

which can answer the question "why user_3408 likes item_318" at a shallow level. To mine more in-depth information from the paths and further differentiate the contribution of each path, we transform these three paths to metapaths by abstracting each entity to an entity type:

- $mp_1 = \text{User} \xrightarrow{Likes} \text{Item} \xrightarrow{BelongsTo} \text{Genre} \xrightarrow{Contains} \text{Item};$
- $mp_2 = \text{User} \xrightarrow{Likes} \text{Item} \xrightarrow{StarredBy} \text{Actor} \xrightarrow{StarsIn} \text{Item};$
- $mp_3 = \text{User} \xrightarrow{Likes} \text{Item} \xrightarrow{StarredBy} \text{Actor} \xrightarrow{Co-starsWith}$ $\text{Actor} \xrightarrow{StarsIn} \text{Item};$

Clearly, these metapaths account for different user habits in the movie recommendation field. mp_1 indicates that the user tends to show interest in movies with the same genre. mp_2 tells us that the user may like movies with the same actors. More interestingly, we can also conclude that user_3408 may be a fan of Morgan Freeman by simple analysis of mp_2. In addition, mp_3 reveals that the user might have some interest in movies co-starring two famous actors. Such correlations between entity types assist us in acquiring more precise user preferences. After an overall statistical analysis by the entropy encoder and the learning process of our model, the weighting scores of the three paths are as shown in Figure 5. Specifically, the score of mp_1 is slightly higher than that of mp_2, while the score of mp_3 is much lower than the other two. These results unambiguously explain that the events "the user likes a movie with the same actor" and "the user likes a movie of the same genre" are more common in movie recommendation. Additionally, the low weight of w_3 demonstrates that the behavior "User Likes Item StarredBy Actor Co-starsWith Actor StarsIn Item" is fairly unusual compared with the other two, which is consistent with our behavioral intuitions in real life.

At last, we give an overall quantitative analysis about these weighting scores, as shown in Figure 7. We extract paths with the greatest weighting score in each user-item interaction. Here the number of the extracted paths is same as the number of user-item interactions. Every path here can be regarded as the most convincing explanation to the recommendation result. Evidently, in KKBox, the number of 3-hop paths are the greatest, which corresponds

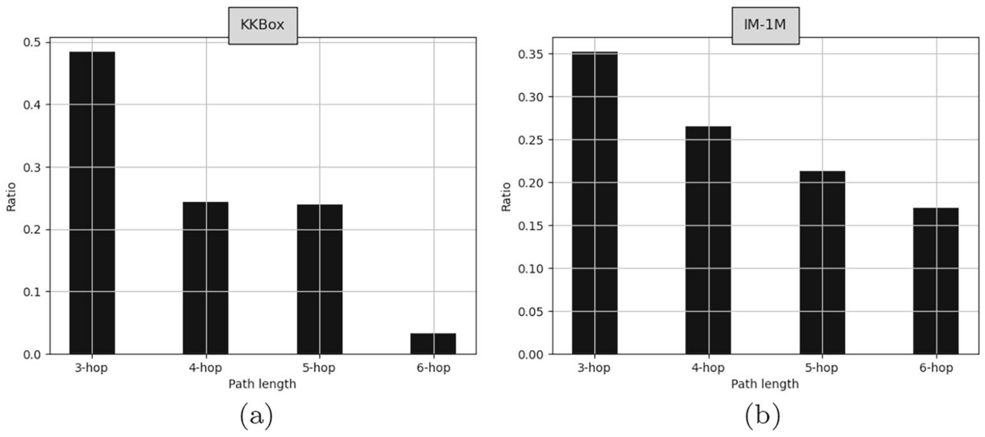


Figure 7 The ratios of the length of path with the greatest weight in user-item interactions. 1-hop and 2-hop paths here do not exist due to our schema graphs in Section 5.1

to a 3-hop path always contributes more to the result. This is a reasonable explanation as the closer relation is always more convincing. In IM-1M, the results are similar to those in KKBox. But number of 6-hop paths has a greater ratio, which indicates, compared to songs, people are more likely to like movies under 6-hop relations. All the above evidence demonstrates that our PeRN has a stronger explainability.

6 Conclusions

In this paper, a path-enhanced recurrent network (PeRN) is proposed, which shows good performance in dealing with cold-start issues and enhances the explanations of KG recommendation. We exploit information from the metapath to compensate for the shortcomings of general path-based methods and introduce a bidirectional path extraction algorithm, since path extraction is time-consuming. Extensive experiments show the well-developed explanations and effectiveness of our method. In the future, we hope to continue our work in the following research directions: (1) We hope to improve the current network model to boost its memory ability. In processing serialized data, such as the paths and metapaths in our PeRN, the memory ability of the model is just as crucial as its learning ability, which is always the most critical part in judging the quality of a model. Current researchers have paid too much attention to RNN-based models such as LSTM and GRU. We believe that memory networks, such as key-value memory networks, can also achieve excellent results, and we hope to leverage them to better remember the path information of KGs. (2) We hope to extract more information from the path. Although we analyze the relationships among different paths by using an entropy-based encoder in PeRN, we think there are more interesting correlations among different paths, such as triangular relationships, multiple relationships and relational reasoning in paths. We will mine this perspective further to improve the utilization of path information and to better excavate user preferences in KG recommendation. (3) In the past two years, research and applications based on graph neural networks (GNNs), such as RecoGCN [29] and GSN [9], have become a new focus. We plan to propose a new GNN-based method for explainable recommendation with KGs.

Acknowledgments This work was supported in part by National Key R&D Program of China under Grants no. 2018YFB1404302, National Natural Science Foundation of China under Grants No.62072203.

References

1. Bordes, A., Glorot, X., Weston, J., Bengio, Y.: A semantic matching energy function for learning with multi-relational data - Application to word-sense disambiguation. Mach. Learn. **94**, 233–259 (2014)
2. Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Proceedings of the 27th Annual Conference on Neural Information Processing Systems, NIPS 2013, pp. 2787–2795. MIT Press, Lake Tahoe (2013)
3. Cao, Y., Wang, X., He, X., Hu, Z., Chua, T.: Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In: Proceedings of The World Wide Web Conference, WWW 2019, pp. 151–161. ACM, San Francisco (2019)
4. Chen, X., Xu, H., Zhang, Y., Tang, J., Cao, Y., Qin, Z., Zha, H.: Sequential recommendation with user memory networks. In: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, pp. 108–116. ACM, Marina Del Rey (2018)
5. Cheng, H., 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.: Wide & deep learning

for recommender systems. In: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, DLRS@RecSys 2016, pp. 7–10. ACM, Boston (2016)
6. Fan, S., Zhu, J., Han, X., Shi, C., Hu, L., Ma, B., Li, Y.: Metapath-guided heterogeneous graph neural network for intent recommendation. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, pp. 2478–2486. ACM, Anchorage (2019)
7. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.: Neural collaborative filtering. In: Proceedings of the 26-th International Conference on World Wide Web, WWW 2017, pp. 173–182. ACM, Perth (2017)
8. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**, 1735–1780 (1997)
9. Hu, W., Chan, Z., Liu, B., Zhao, D., Ma, J., Yan, R.: GSN: A graph-structured network for multi-party dialogues. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, pp. 5010–5016. Morgan Kaufmann, Macao (2019)
10. Huang, X., Fang, Q., Qian, S., Sang, J., Li, Y., Xu, C.: Explainable interaction-driven user modeling over knowledge graph for sequential recommendation. In: Proceedings of the 27th ACM International Conference on Multimedia, MM 2019, pp. 548–556. ACM, Nice (2019)
11. Huang, J., Zhao, W., Dou, H., Wen, J., Chang, E.: Improving sequential recommendation with knowledge-enhanced memory networks. In: Proceedings of The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, pp. 505–514. ACM, Ann Arbor (2018)
12. Ji, G., He, S., Xu, L., Liu, K., Zhao, J.: Knowledge graph embedding via dynamic mapping matrix. In: Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language, ACL 2015, pp. 687–696. ACL, Beijing (2015)
13. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI 2015, pp. 2181–2187. AAAI, Austin (2015)
14. Meng, Y., Rumshisky, A., Romanov, A.: Temporal information extraction for question answering using syntactic dependencies in an LSTM-based architecture. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, pp. 887–896, Copenhagen (2017)
15. Nickel, M., Tresp, V., Kriegel, H.: A three-way model for collective learning on multi-relational data. In: Proceedings of the 28th International Conference on Machine Learning, ICML 2011, pp. 809–816. ACM, Bellevue (2011)
16. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian personalized ranking from implicit feedback. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI 2009, pp. 452–461. AUAI, Montreal (2009)
17. Roy, P., Boddeti, V.: Mitigating information leakage in image representations: a maximum entropy approach. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, pp. 2586–2594. IEEE, Long Beach (2019)
18. 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**, 992–1003 (2011)
19. Sun, Z., Yang, J., Zhang, J., Bozzon, A., Huang, L., Xu, C.: Recurrent knowledge graph embedding for effective recommendation. In: Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, pp. 297–305. ACM, Vancouver (2018)
20. Wan, W., Chen, J., Li, T., Huang, Y., Tian, J., Yu, C., Xue, Y.: Information entropy based feature pooling for convolutional neural networks. In: Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, pp. 3404–3413. IEEE, Seoul (2019)
21. Wang, X., He, W., Cao, Y., Liu, M., Chua, T.: KGAT: Knowledge graph attention network for recommendation. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, pp. 950–958. ACM, Anchorage (2019)
22. Wang, X., He, X., Wang, M., Feng, F., Chua, T.: Neural graph collaborative filtering. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, pp. 165–174. ACM, Paris (2019)
23. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: a survey of approaches and applications. IEEE Trans. Knowl. Data Eng. **29**, 2724–2743 (2017)
24. Wang, X., Wang, D., Xu, C., He, X., Cao, Y., Chua, T.: Explainable reasoning over knowledge graphs for recommendation. In: Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, pp. 5329–5336. AAAI, Honolulu (2019)
25. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2014, pp. 1112–1119. AAAI, Québec City (2014)
26. Wang, H., Zhang, F., Wang, J., Zhao, M., Li, W., Xie, X., Guo, M.: RippleNet: Propagating user preferences on the knowledge graph for recommender systems. In: Proceedings of the 27th ACM International

Conference on Information and Knowledge Management, CIKM 2018, pp. 417–426. ACM, Torino (2018)

27. Wang, H., Zhang, F., Xie, X., Guo, M.: DKN: Deep Knowledge-aware network for news recommendation. In: Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, pp. 1835–1844. ACM, Lyon (2018)
28. Xiao, J., Ye, H., He, X., Zhang, H., Wu, F., Chua, T.: Attentional factorization machines: Learning the weight of feature interactions via attention networks. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, pp. 3119–3125. Morgan Kaufmann, Melbourne (2017)
29. Xu, F., Lian, J., Han, Z., Li, Y., Xu, Y., Xie, X.: Relation-aware graph convolutional networks for agent-initiated social e-commerce recommendation. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, pp. 529–538. ACM, Beijing (2019)
30. Xu, Y., Zhu, Y., Shen, Y., Yu, J.: Learning shared vertex representation in heterogeneous graphs with convolutional networks for recommendation. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, pp. 4620–4626. Morgan Kaufmann, Macao (2019)
31. Yu, X., Ren, X., Sun, Y., Gu, Q., Sturt, B., Khandelwal, U., Norick, B., Han, J.: Personalized entity recommendation: a heterogeneous information network approach. In: Seventh ACM International Conference on Web Search and Data Mining, WSDM 2014, pp. 283–292. ACM, New York (2014)
32. Zhang, F., Yuan, N., Lian, D., Xie, X., Ma, W.: Collaborative knowledge base emedding for recommender systems. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2016, pp. 353–362. ACM, San Francisco (2016)
33. Zhao, H., Yao, Q., Li, J., Song, Y., Lee, D.: Meta-graph based recommendation fusion over heterogeneous information networks. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2017, pp. 635–644. ACM, Halifax (2017)
34. Zheng, Y., Gao, C., He, X., Li, Y., Jin, D.: Price-aware recommendation with graph convolutional networks. In: Proceedings of the 36th IEEE International Conference on Data Engineering, ICDE 2020, pp. 133–144. IEEE, Dallas (2020)