Explainable Reasoning over Knowledge Graphs for Recommendation

Xiang Wang[1]*, Dingxian Wang[2]†, Canran Xu[2], Xiangnan He[1,3], Yixin Cao[1], Tat-Seng Chua[1]
[1]School of Computing, National University of Singapore, [2]eBay
[3]School of Information Science and Technology, University of Science and Technology of China
xiangwang1223@gmail.com, {diwang, canxu}@ebay.com, {xiangnanhe, caoyixin2011}@gmail.com, dcscts@nus.edu.sg

Abstract

Incorporating knowledge graph into recommender systems has attracted increasing attention in recent years. By exploring the interlinks within a knowledge graph, the connectivity between users and items can be discovered as paths, which provide rich and complementary information to user-item interactions. Such connectivity not only reveals the semantics of entities and relations, but also helps to comprehend a user's interest. However, existing efforts have not fully explored this connectivity to infer user preferences, especially in terms of modeling the sequential dependencies within and holistic semantics of a path.

In this paper, we contribute a new model named *Knowledge-aware Path Recurrent Network* (KPRN) to exploit knowledge graph for recommendation. KPRN can generate path representations by composing the semantics of both entities and relations. By leveraging the sequential dependencies within a path, we allow effective reasoning on paths to infer the underlying rationale of a user-item interaction. Furthermore, we design a new weighted pooling operation to discriminate the strengths of different paths in connecting a user with an item, endowing our model with a certain level of explainability. We conduct extensive experiments on two datasets about movie and music, demonstrating significant improvements over state-of-the-art solutions *Collaborative Knowledge Base Embedding* and *Neural Factorization Machine*.

Introduction

Prior efforts have shown the importance of incorporating auxiliary data into recommender systems, such as user profiles (Wang et al. 2018c) and item attributes (Bayer et al. 2017). Recently, knowledge graphs (KGs) have attracted increasing attention (Zhang et al. 2016; Shu et al. 2018; Wang et al. 2018a), due to its comprehensive auxiliary data: background knowledge of items and their relations amongst them. It usually organizes the facts of items in the form of triplets like (*Ed Sheeran*, *IsSingerOf*, *Shape of You*), which can be seamlessly integrated with user-item interactions (Chaudhari, Azaria, and Mitchell 2016; Cao et al. 2017). More important, by exploring the interlinks within a KG, the connectivity between users and items reflects their underlying relationships, which are complementary to user-item interaction data.

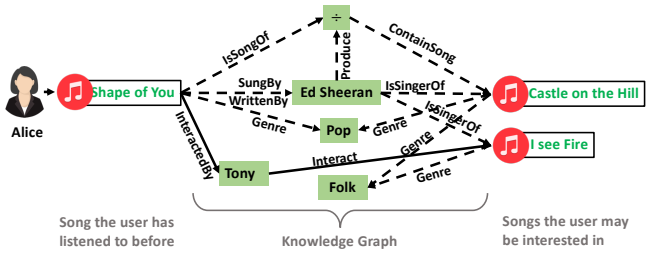


Figure 1: Illustration of KG-aware recommendation in the music domain. The dashed lines between entities are the corresponding relations, while the sold lines are the user-item interactions.

Extra user-item **connectivity** information derived from KG endows recommender systems the ability of **reasoning** and **explainability**. Taking music recommendation as an example (Figure 1), a user is connected to *I See Fire* since she likes *Shape of You* sung by the same singer *Ed Sheeran*. Such connectivity helps to **reason** about unseen user-item interactions (*i.e.,* a potential recommendation) by synthesizing information from paths.

Running Example: (*Alice, Interact, Shape of You*)∧(*Shape of You, SungBy, Ed Sheeran*)∧(*Ed Sheeran, IsSingerOf, I See Fire*)⇒(*Alice, Interact, I See Fire*).

Clearly, the reasoning unveils the possible user intents behind an interaction, offering **explanations** behind a recommendation. How to model such connectivity in KGs, hence, is of critical importance to inject knowledge into a recommender systems.

Prior efforts on knowledge-aware recommendation are roughly categorized into path and embedding fashion. Path-based methods (Yu et al. 2014; 2013; Gao et al. 2018) introduce *meta-paths* to refine the similarities between users and items. However, we argue that meta-path is inefficient in reasoning over KGs, owing to the following limitations: 1) As relations are usually excluded from meta-paths, they hardly specify the holistic semantics of paths, especially when similar entities but different relations are involved in a meta-path; and 2) They fail to automatically uncover and reason on unseen connectivity patterns, since meta-paths

*The first three authors have equal contribution.
†Dingxian Wang is the corresponding author.

requires domain knowledge to be predefined.

Another line of research (Zhang et al. 2016; Wang et al. 2018b; Huang et al. 2018) leverages knowledge graph embedding (KGE) techniques, such as TransE (Bordes et al. 2013) and TransR (Lin et al. 2015), to regularize the representations of items. As a result, items with similar connected entities have similar representations, which facilitate the collaborative learning of user interests. Despite performance improvements, we argue that KGE regularization lacks the reasoning ability. Specially, it only considers direct relations between entities, rather than the multi-hop relation paths as the Running Example shows. Moreover, the characterization of user-item connectivity is achieved in a rather implicit way, that is, to guide the representation learning, but not to infer a user's preference.

In this work, we aim to fill the research gap by developing a solution that reasons on paths to infer user preferences on items. In terms of reasoning, we expect our method to model the sequential dependencies of entities and sophisticated relations of a path connecting a user-item pair. In terms of explainability, we would like our method to discriminate the different contributions of different paths, when inferring user interests.

Towards this end, we propose a new solution, named Knowledge-aware Path Recurrent Network (KPRN), which not only generates representations for paths by accounting for both entities and relations, but also performs reasoning based on paths to infer user preference. Specifically, we first extract qualified paths between a user-item pair from the KG, each of which consists of the related entities and relations. We then adopt a Long Short-Term Memory (LSTM) network to model the sequential dependencies of entities and relations. Thereafter, a pooling operation is performed to aggregate the representations of paths to obtain prediction signal for the user-item pair. More importantly, the pooling operation is capable of discriminating the contributions of different paths for a prediction, which functions as the attention mechanism (Chen et al. 2017; Neelakantan, Roth, and McCallum 2015). Owing to such attentive effect, our model can offer path-wise explanations such as *Castle on the Hill is recommended since you have listened to Shape of You sung and written by Ed Sheeran*. We conduct extensive experiments on two datasets to verify our method.

The contributions of this work are threefold:

- We highlight the importance of performing explicit reasoning on KG to better reveal the reasons behind a recommendation.
- We propose an end-to-end neural network model to learn path semantics and integrate them into recommendation.
- We contribute a dataset to study KG for recommendation by aligning a MovieLens benchmark with IMDB. We verify our method on the data, and release the data and the codes to facilitate the community working on emerging field of KG-enhanced recommendation.

Knowledge-aware Path Recurrent Network

In this section, we elaborate our proposed method, as illustrated in Figure 2. Before introducing our proposed method, we first formally define Knowledge Graph, user-item data and describe how to combine them in an enriched knowledge graph as the inputs of our model.

Background

A knowledge Graph (KG) is a directed graph whose nodes are entities $\mathcal{E}$ and edges $\mathcal{R}$ denote their relations. Formally, we define KG as $\mathcal{KG} = \{(h, r, t)|h, r \in \mathcal{E}, r \in \mathcal{R}\}$, where each triplet (h, r, t) indicates a fact that there is a relationship r from head entity h to tail entity t.

The user-item interaction data is usually presented as a bipartite graph. In particular, we use $\mathcal{U} = \{u_t\}_{t=1}^{M}$ and $\mathcal{I} = \{i_t\}_{t=1}^{N}$ to separately denote the user set and the item set, where M and N are the number of users and items, respectively. Following (Chaudhari, Azaria, and Mitchell 2016), we represent the interaction between a user and an item with a triplet τ =(u, *interact*, i), if there is an observed interaction (*e.g.,* rate, click, and view feedbacks), where *interact* is a pre-defined relation.

We merge the item set and the entity set through string matching: $\mathcal{I} \subseteq \mathcal{E}$, so that the two structural data are integrated into an enriched knowledge graph $\mathcal{G} = \{(h, r, t)|h, r \in \mathcal{E}', r \in \mathcal{R}'\}$, where $\mathcal{E}' = \mathcal{E} \cup \mathcal{U}$ and $\mathcal{R}' = \mathcal{R} \cup \{\text{interact}\}$. For consistency, the Knowledge Graph (KG) in the rest paper denotes the combined graph $\mathcal{G}$ including both original KG and user-item data, otherwise noted.

Preference Inference via Paths

The triplets in the KG clearly describe direct or indirect (i.e. multiple-step) relational properties of items, which shall constitute one or several paths between the given user and item pair. We explore these paths in order to achieve comprehensively reasoning and understanding for recommendation.

Within $\mathcal{G}$, we formally define the path from the user u to the item i as a sequence of entities and relations: $p = [e_1 \xrightarrow{r_1} e_2 \xrightarrow{r_2} \cdots \xrightarrow{r_{L-1}} e_L]$, where $e_1 = u$, $e_L = i$; (e_l, r_l, e_{l+1}) is the l-th triplet in p, and L denotes the number of triplets in the path. The construction of paths will be elaborated in the section of Path Extraction.

Next, we will use a realistic example to show the sophisticated relations (i.e. paths) between a user and an item behind their possible interactions, which inspires us to model the high-level semantics of path compositionally by considering both entities and (multiple-step) relations.

Examples: Consider the music recommendation shown in Figure 1, where the "listen to Castle on the Hill" behavior of user Alice can be referred by the following paths:

- p_1 = [Alice $\xrightarrow{\text{Interact}}$ Shape of You $\xrightarrow{\text{IsSongOf}}$ $\div$ $\xrightarrow{\text{ContainSong}}$ Castle on the Hill];
- p_2 = [Alice $\xrightarrow{\text{Interact}}$ Shape of You $\xrightarrow{\text{SungBy}}$ Ed Sheeran $\xrightarrow{\text{IsSingerOf}}$ Castle on the Hill].

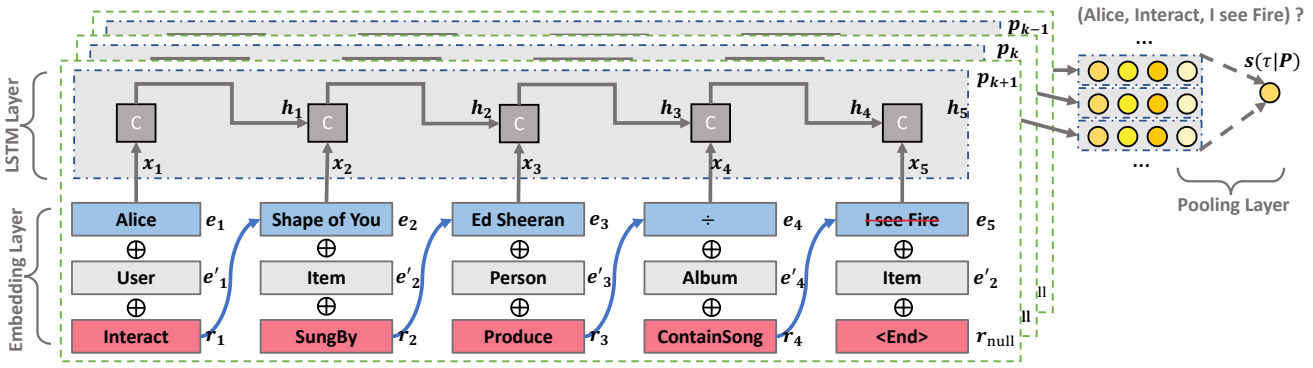


Figure 2: Schematic overview of our model architecture. The embedding layer contains 3 individual layers for entity, entity type, and relation type, respectively. The concatenation of the 3 embedding vectors is the input of LSTM for each path.

- p_3 = [Alice $\xrightarrow{\text{Interact}}$ Shape of You $\xrightarrow{\text{InteractedBy}}$ Tony $\xrightarrow{\text{Interact}}$ Castle on the Hill];

These paths from the same user *Alice* to the same item *Castle on the Hill* obviously express their different multiple-step relations, and implies various compositional semantics and possible explanations of the listen behavior. In particular, p_1 and p_2 infer that *Alice* may prefer songs that belonging to the album ÷ and the songs sung by *Ed Sheeran*, while p_3 reflects the collaborative filtering (CF) effect: similar users tend to have similar preferences. Therefore, from the view of reasoning, we consume the connectivity along all paths to learn compositional relation representations, and weighted pool them together for predicting the *interact* relation between the user and the target item.

Task Definition: Our task can be formulated as follows: given a user u, a target item i, and a set of paths $\mathcal{P}(u,i) = \{p_1, p_2, \cdots, p_K\}$ connecting u and i, the holistic goal is to estimate the interaction by:

$$\hat{y}_{ui} = f_\Theta(u, i|\mathcal{P}(u,i)), \tag{1}$$

where f denotes the underlying model with parameters Θ, and $\hat{y}_{ui}$ presents the predicted score for the user-item interaction. Distinct from embedding-based methods, we can explain $\hat{y}_{ui}$ as the plausibility score of the triplet $\tau = (u, \textit{interact}, i)$ inferred by the connectivity $\mathcal{P}(u,i)$.

Modeling

KPRN takes a set of paths of each user-item pair as input, and outputs a score indicating how possible the user will interact the target item. As illustrated in Figure 2, there are three key components: (1) embedding layer to project three types of IDs information: the entity, entity type, and the relation pointing to the next node into a latent space, (2) LSTM layer that encodes the elements sequentially with the goal of capturing the compositional semantics of entities conditioned on relations, and (3) pooling layer to combine multiple paths and output the final score of the given user interacting the target item.

Embedding Layer Given a path p_k, we project the type (*e.g.,* person or movie) and specific value (*e.g.,* Peter Jackson or The Hobbit II) of each entity into two separate embedding vectors, $\mathbf{e}_l \in \mathbb{R}^d$ and $\mathbf{e}'_l \in \mathbb{R}^d$, where d is the embedding size.

In real-world scenarios, it is common that the same entity-entity pairs may have different semantics due to different relations connecting them. Such differences may reveal the diverse intents about why a user selected the item. As an example, let (*Ed Sheeran, IsSingerOf, Shape of You*) and (*Ed Sheeran, IsSongwriterOf, Shape of You*) be the triplets in two paths referring a user's preferences. Without specifying the relations, these paths will be represented as the same embeddings, regardless of the possibility that the user only prefers songs sung by Ed Sheeran, rather than that written by Ed Sheeran. We hence believe that it is important to explicitly incorporate the semantics of relations into path representation learning. Towards this end, each relation r_l in p_k is represented as an embedding vector $\mathbf{r}_l \in \mathbb{R}^d$. As a result, we obtain a set of embeddings for path p_k, $[\mathbf{e}_1, \mathbf{r}_1, \mathbf{e}_2, \cdots, \mathbf{r}_{L-1}, \mathbf{e}_L]$, where each element denotes an entity or a relation.

LSTM Layer With the embedding sequence to describe a path, we can employ RNN models to explore the sequential information, and generate a single representation for encoding its holistic semantics. Among various RNN methods, we adopt LSTM since it is capable of memorizing long-term dependency in a sequence. Such long-term sequential pattern is crucial to reason on paths connecting a user and item entities to estimate the confidence of the "interact" relation.

At the path-step $l-1$, the LSTM layer outputs a hidden state vector $\mathbf{h}_{l-1}$, consuming the subsequence $[e_1, r_1, \cdots, e_{l-1}, r_{1-1}]$. Simultaneously, we concatenate the embedding of current entity e_{l-1} and relation r_{l-1} as the input vector:

$$\mathbf{x}_{l-1} = \mathbf{e}_{l-1} \oplus \mathbf{e}'_{l-1} \oplus \mathbf{r}_{l-1}, \tag{2}$$

where $\oplus$ is the concatenation operation. Noted that, for the last entity e_L, a null relation r_L is padded to the end of path. As such, the input vector contains not only the sequential information, but also the semantic information of the entity and its relation to the next entity. Consequently, $\mathbf{h}_{l-1}$ and $\mathbf{x}_{l-1}$ are used to learn the hidden state of the next path-step

l, which is defined via the following equations:

$$\begin{aligned}
\mathbf{z}_l &= \tanh(\mathbf{W}_z\mathbf{x}_l + \mathbf{W}_h\mathbf{h}_{l-1} + \mathbf{b}_z) \\
\mathbf{f}_l &= \sigma(\mathbf{W}_f\mathbf{x}_l + \mathbf{W}_h\mathbf{h}_{l-1} + \mathbf{b}_f) \\
\mathbf{i}_l &= \sigma(\mathbf{W}_i\mathbf{x}_l + \mathbf{W}_h\mathbf{h}_{l-1} + \mathbf{b}_i) \\
\mathbf{o}_l &= \sigma(\mathbf{W}_o\mathbf{x}_l + \mathbf{W}_h\mathbf{h}_{l-1} + \mathbf{b}_o) \\
\mathbf{c}_l &= \mathbf{f}_l \odot \mathbf{c}_{l-1} + \mathbf{i}_l \odot \mathbf{z}_l \\
\mathbf{h}_l &= \mathbf{o}_l \odot \tanh(\mathbf{c}_l)
\end{aligned} \tag{3}$$

where $\mathbf{c}_l \in \mathbb{R}^{d'}$, $\mathbf{z} \in \mathbb{R}^{d'}$ denote the cell (memory) state vector and information transform module, respectively, and d' is the number of hidden units; $\mathbf{i}_l$, $\mathbf{o}_l$, and $\mathbf{f}_l$ separately represents the input, output, and forget gate. $\mathbf{W}_z$, $\mathbf{W}_i$, $\mathbf{W}_f$, and $\mathbf{W}_o \in \mathbb{R}^{d' \times 3d}$, and $\mathbf{W}_h \in \mathbb{R}^{d' \times d'}$ are mapping coefficient matrices, while $\mathbf{b}_z$, $\mathbf{b}_i$, $\mathbf{b}_f$, and $\mathbf{W}_o$ are bias vectors. $\sigma(\cdot)$ is the activation function set as sigmoid, and $\odot$ stands for the element-wise product of two vectors. Taking advantages of the memory state, the last state $\mathbf{h}_L$ is capable of representing the whole path $\mathbf{p}_k$.

Having established the representation of path $\mathbf{p}_k$, we aim to predict the plausibility of $\tau = (u, \text{interact}, i)$. Towards this end, two fully-connected layers are adopted to project the final state into the predictive score for output, given by:

$$s(\tau|\mathbf{p}_k) = \mathbf{W}_2^\top \text{ReLU}(\mathbf{W}_1^\top \mathbf{p}_k), \tag{4}$$

where $\mathbf{W}_1$ and $\mathbf{W}_2$ are the coefficient weights of the first and second layers respectively, bias vectors are omitted form simplicity, and the rectifier is adopted as the activation function.

Weighted Pooling Layer A user-item entity pair usually has a set of paths connecting them in a KG. Let $\mathcal{S} = \{s_1, s_2, \cdots, s_K\}$ be the predictive scores for K paths, $\mathcal{P}(u, i) = \{p_1, p_2, \cdots, p_K\}$, connecting a user-item pair (u, i), where each element is calculated based on Equation (4). The final prediction could be the average of the scores of all paths, which is formulated as,

$$\hat{y}_{ui} = \sigma(\frac{1}{K}\sum_{k=1}^{K} s_k). \tag{5}$$

Nevertheless, prior studies (Yu et al. 2014; Shu et al. 2018) suggest that different paths have varying contributions to model user preferences, while Equation (5) fails to specify importance of each path. Inspired by previous work (McCallum et al. 2017; Chen et al. 2017), we design a weighted pooling operation to aggregate scores of all paths. Here the pooling function is defined as follows,

$$g(s_1, s_2, \cdots, s_K) = \log\left[\sum_{k=1}^{K} \exp\left(\frac{s_k}{\gamma}\right)\right], \tag{6}$$

and the final prediction score is given by,

$$\hat{y}_{ui} = \sigma(g(s_1, s_2, \cdots, s_K)), \tag{7}$$

where γ is the hyper-parameter to control each exponential weight. Such pooling is capable of distinguishing the path importance, which is attributed by the gradient:

$$\frac{\partial g}{\partial s_k} = \frac{\exp(s_k/\gamma)}{\gamma \sum_{k'} \exp(s_{k'}/\gamma)}, \tag{8}$$

which is proportional to the score of each path during the back-propagation step. Moreover, the pooling function endows the final prediction more flexibility. In particular, when setting $\gamma \to 0$, the pooling function can degenerate to max-pooling; whereas, it can degrade to mean-pooling by setting $\gamma \to \infty$. We conduct a case study on exploring the utility of the weighted pooling operation in Section Case Studies.

Learning

Similar to the spirit in recent work (He et al. 2017; 2018; Wang et al. 2017), we treat the recommender learning task as a binary classification problem, where an observed user-item interaction is assigned a target value 1, otherwise 0. We use the pointwise learning methods to learn the parameters of our model. In particular, the negative log-likelihood is adopted as the objective function, which is defined as follows,

$$\mathcal{L} = -\sum_{(u,i)\in\mathcal{O}^+} \log \hat{y}_{ui} + \sum_{(u,j)\in\mathcal{O}^-} \log(1 - \hat{y}_{uj}), \tag{9}$$

where $\mathcal{O}^+ = \{(u, i)|y_{ui} = 1\}$ and $\mathcal{O}^- = \{(u, j)|y_{uj} = 0\}$ are the positive and negative user-item interaction pairs, respectively. We conduct L_2 regularization on the trainable parameters Θ, which is omitted here for simplicity, to avoid overfitting. We elaborate the implementation details in the section of Experimental Settings.

Experiments

In this section, we perform experiments on two real-world datasets to evaluate our proposed method. We aim to answer the following research questions:

- **RQ1:** Compared with the state-of-the-art KG-enhanced methods, how does our method perform?
- **RQ2:** How does the multi-step path modeling (*e.g.,* the incorporation of both entity and relation types) affect KPRN?
- **RQ3:** Can our proposed method reason on paths to infer user preferences towards items?

Dataset Description

We consider two scenarios: movie recommendation and music recommendation. For movie domain, we use the combination of MovieLens-1M[1] and IMDb[2] datasets, named MI, which are linked by the titles and release dates of movies. In particular, MovieLens-1M offers the user-item interaction data, while IMDb serves as the KG part that contains auxiliary information on movies, such as genre, actor, director, and writer. For music domain, we use the benchmark dataset, KKBox, which is adopted from the WSDM cup 2018 Challenge[3] and is provided by the music streaming service KKBox. Beyond the user-item interaction data, this dataset contains the descriptions of music like singer, songwriter, and genre. The statistics of two datasets are summarized in Table 1.

[1] https://grouplens.org/datasets/movielens/1m/.
[2] https://www.imdb.com/.
[3] https://wsdm-cup-2018.kkbox.events/.

Table 1: Statistics of our datasets.

	Dataset	MI	KKBox
User-Item Interaction	#Users	6,040	34,403
	#Items	3,859	2,296,833
	#Interactions	998,034	3,714,655
Knowledge Graph	#Entities	11,462	2,851,220
	#Entity Types	4	4
	#Relation Types	6	6
	#Triplets	1,017,030	11,182,682
Path	#Paths	55,573,556	38,192,484
	Avg Path Length	5.07	5.09

Following previous efforts (Yu et al. 2014; He et al. 2017; Shu et al. 2018), we process the datasets as: if a user rates a movie or has an interaction record with a song, we set the user-movie or user-song pair as the observed positive feedback with the target value of 1, and 0 otherwise.

For each dataset, we holdout the 80% and 20% interaction history of each user randomly to construct the training and test sets. For each positive user-item interaction pair in the training set, we conducted the negative sampling strategy to pair it with four negative items that the user has not interacted with. During the test stage, the ratio between positive and negative interactions is set as $1:100$, namely, 100 negative items are randomly sampled and pair with one positive item in the testing set.

Path Extraction

In practice, it is labor intensive and infeasible to fully exploring all connected paths over the KG. Especially, the number of paths grows exponentially *w.r.t.* the length of path, where millions of interlinks will be generated. As suggested in prior efforts (Sun et al. 2011; Shu et al. 2018), truncating all paths at a certain length and disregarding remote connections are sufficient to model the connectivity between a user-item pair. Moreover, as pointed out by (Sun et al. 2011), paths with length greater than six will introduce noisy entities. Therefore, we extract all qualified paths, each with length up to six, that connect all user-item pairs.

Experimental Settings

Evaluation Metrics We adopt two evaluation protocols to evaluate the performance of top-K recommendation and preference ranking, respectively, given by:

- **hit**@K considers whether the relevant items are retrieved within the top K positions of the recommendation list.
- **ndcg**@K measures the relative orders among positive and negative items within the top K of the ranking list.

We report the average metrics at $K = \{1, 2, \cdots, 15\}$ of all instances in the test set.

Baselines We compare our proposed method with the following methods:

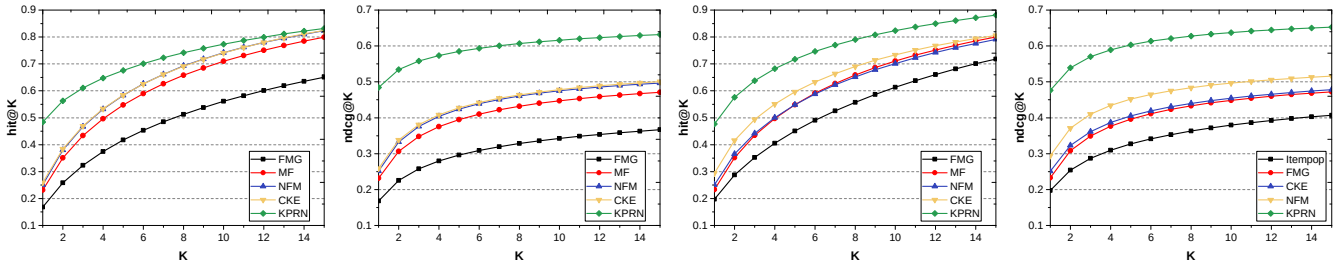
- **MF** (Rendle et al. 2009): This is matrix factorization with Bayesian personalized ranking (BPR) loss, which solely utilizes user-item interaction.
- **NFM** (He and Chua 2017): The method is a state-of-the-art factorization model which treats historical items as the features of users. Specially, we employed one hidden layer as suggested in (He and Chua 2017).
- **CKE** (Zhang et al. 2016): Such embedding-based method is tailored for KG-enhanced recommendation, which integrates the representations from Matrix Factorization (Rendle et al. 2009) and TransR (Lin et al. 2015) to enhance the recommendation.
- **FMG** (Zhao et al. 2017): This is a state-of-the-art meta-path based method, which predefines various types of meta-graphs and employs Matrix Factorization on each meta-graph similarity matrix to make recommendation.

Parameter Settings For fair comparison, we learn all models from scratch without any pretrained parameters. We optimize all models with Adaptive Moment Estimation (Adam) and apply a grid search to find out the best settings of hyperparameters. The learning rate is searched in $\{0.001, 0.002, 0.01, 0.02\}$, while the coefficient of L_2 regularization is tuned amongst $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$. Other hypermeters of our proposed model are empirically set as follows: the batch size is 256, the embedding size of relation and entity type is 32, the embedding size of entity value is 64, and the unit number of LSTM is 256. The dimensions of latent factors for MF, NFM, and CKE are empirically set to be 64. For FMG, we set the rank used to factorize meta-graph similarity matrices to be 10, and the factor size of the second-order weights as 10, as suggested by (Zhao et al. 2017). Moreover, the early stopping strategy is performed, *i.e.,* premature stopping if hit@15 on the test data does not increase for five successive epochs.

Performance Comparison (RQ1)

Figure 3 reports our experimental results on two datasets *w.r.t.* hit@K and ndcg@K. We have the following findings:

- FMG gives poor performance in both datasets. This indicates that meta-graph based methods, which rely heavily on the predefined meta-graph patterns, may introduce remote entities and fail to fully explore the user-item connectivity.
- NFM achieves better performance than MF. It makes sense since by treating the rated items as the user features, NFM essentially enhances the second-order user-item proximity ,while MF only considers the first-order user-item connections.
- Compared to CF-based methods (MF and NFM), the performance of CKE indicates that incorporating KG can solve the data sparsity issue effectively. In particular, CKE shows consistent improvement over KKBox dataset that is extremely sparse, while only achieving comparable performance to NFM on MI dataset which has denser interaction data.
- KPRN substantially outperforms CKE *w.r.t.* hit@K and ndcg@K, achieving the best performance. By leveraging paths to infer user preference, KPRN is capable of exploring the user-item connectivity in an explicit way,

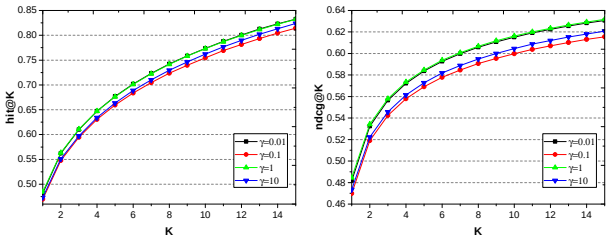


(a) hit@K on MI (b) ndcg@K on MI (c) hit@K on KKBox (d) ndcg@K on KKBox

Figure 3: Top-K recommendation performance between all the methods on MI and KKBox datasets *w.r.t.* hit@K and ndcg@K, where $K = \{1, 2, \cdots, 15\}$.

Table 2: Performance comparison of KPRN and KPRN-r and their effects on relation modeling.

	MI						KKBox					
	hit@5	hit@10	hit@15	ndcg@5	ndcg@10	ndcg@15	hit@5	hit@10	hit@15	ndcg@5	ndcg@10	ndcg@15
KPRN-r	0.635	0.738	0.801	0.533	0.566	0.583	0.712	0.821	0.878	0.607	0.632	0.647
KPRN	0.676	0.773	0.832	0.584	0.616	0.632	0.717	0.823	0.881	0.613	0.637	0.652



(a) hit@K on MI (b) ndcg@K on MI

Figure 4: Performance comparison of KPRN *w.r.t.* γ on the MI dataset.

while the embedding-based method (CKE) only utilizes KG to guide the representation learning of items. This verifies the importance of leveraging both entities and relations of KG. Further analyzing Figures 3(b) and 3(d) reveal that KPRN outperforms other baselines by a larger margin *w.r.t.* ndcg@K, demonstrating the strong capacity of preference ranking.

Study of KPRN (RQ2)

To investigate the role of path modeling, we start by explore the influence of relation in paths. We then study how the weighted pooling operation affects the performance.

Effects of Relation Modeling We consider one variant of KPRN without the relation modeling, termed as KPRN-r. In particular, the relation embedding $\mathbf{r}_{l-1}$ in Equation (2) is discarded to generate the input vector $\mathbf{x}_{l-1}$. In Table 2, we compare KPRN with KPRN-r in terms of hit@K and ndcg@K, where K is selected from $\{5, 10, 15\}$. We have the following observations:

- Without considering relations in paths, the performance of KPRN-r decreases on both datasets. This justifies our intuition that specifying different relations is of importance to capture the path semantics, especially when the same entities are involved.
- We find that KPRN improves KPRN-r by 6.45% *w.r.t.* hit@5 on MI, while only 0.70% on KKBox. One reason may be that as MI is much denser than KKBox and it is common that, in MI, multiple paths connect a user-item pair with similar entities but different relations, whereas fewer paths are offered in KKBox. This demonstrates that, given strong connectivity between users and items, specifying relations of paths is of more importance to explore the fine-grained interests of users.

Effects of Weighted Pooling To integrate the prediction scores of multiple paths between a user-item pair, a weighted pooling operation is carefully designed. To analyze its effect, we set the value γ as $\{0.01, 0.1, 1, 10\}$ and report the performance on MI in Figure 4. We find that,

- When γ decrease from 1 to 0.1, the weighted pooling operation degrades the performance, since it is similar to max-pooling and selects only the most important paths as the user-item connectivity.
- The performance *w.r.t.* hit@K and ndcg@K becomes poorer, when increasing γ from 1 to 10. It makes sense since it tends to aggregate contributions from more paths, rather than the most informative ones.

Case Studies (RQ3)

Another desirable property of KPRN is to reason on paths to infer the user preferences towards target items and generate reasonable explanations. This is because our model capture the higher-level semantics from these key factors: entity, entity type, and relation. To demonstrate this, we show an example drawn from KPRN on movie recommendation task.

We randomly select a user, whose ID is u4825 in MovieLens-1M, and select the movie Shakespeare in Love from her interaction record. We then extract all the qualified paths connecting the user-item pair and present the subgraph in Figure 5. We have several observations.

- Collaborative filtering effect plays a pivotal rule to recommend the movie Shakespeare in Love to the user, since the interaction behaviors from other users (*e.g.*, u940 and u5448) are involved in two paths. In particular, the path containing u5448 offers the high contribution score of 0.356 to infer the user's interest.

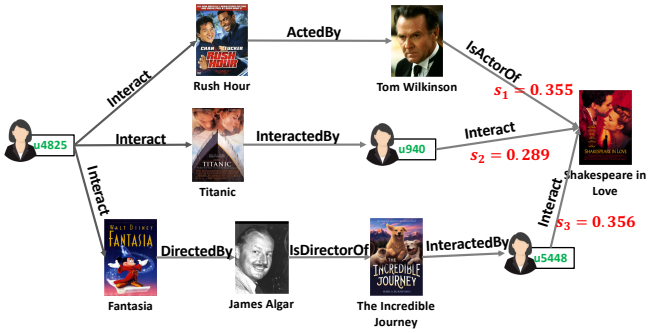


Figure 5: Visualization of three paths with prediction scores for the user of u4825 in MI dataset. The prediction scores are normalized for illustration.

- The target item is connected to what u4825 has watched before (*e.g., Rush Hour, Titanic, and Fantasia*) by the shared knowledge entities, such as actor (*Tom Wilkinson*) and director (*James Algar*). This shows that KPRN is capable of extending user interests along KG paths.
- Analyzing these three paths jointly, we find that different paths describe the user-item connectivity from dissimilar angles, which can be treated as the evidence why the item is suitable for the user. Specially, we can offer path-wise explanations such as *Shakespeare in Love is recommended since you have watched Rush Hour acted by the same actor Tom Wilkinson* or *since it is similar to Titanic that you watched before.* This case demonstrates KPRN's capacity of providing informative explanations.

Related Work

Previous solutions on integrating KG into recommendation can be roughly categorized into embedding-based and path-based methods.

Embedding-based Methods

Prior efforts (Nickel, Tresp, and Kriegel 2011; Zhang et al. 2016; Bellini et al. 2017; Wang et al. 2018b) leverage knowledge graph embedding techniques to guide the representation learning of items. For example, for each item, Zhang *et al.* (2016) generated the representation by combining its latent factor from MF and its semantic embedding from TransR. When performing news recommendation, Wang *et al.* (2018b) generated the news representation by integrating the knowledge-aware embeddings and word embedding of each word entity within the news content. More recently, Huang *et al.* (2018) adopted TransE (Bordes et al. 2013) to generate representations for entities and items, and employed memory networks to update the user representations based on her preferences on specific entities. By exploiting the KG to guide the representation learning, such methods achieve significant improvement in performance. However, we argue that KGE regularization has not fully explored the connectivity between users and items. One reason is that the characterization of user-item connectivity is achieved in a rather implicit way. Moreover, they lack the reasoning ability to infer why a item is recommended for a user.

Path-based Methods

In the literature of path-based methods, some prior studies introduce the connectivity patterns, termed as meta-paths, to explicitly guide the recommender leaning (Heitmann and Hayes 2010; Sun and Han 2012; Yu et al. 2013; 2014; Shi et al. 2015; Gao et al. 2018; 2018). Meta-path is defined as a sequence of entity type, such as user-movie-direct-movie, to capture the user-item affinities carried in KG. For instance, Yu *et al.* (2014) conducted MF framework over meta-path similarity matrices to perform recommendation. Such methods, however, use the user-item connectivity to update user-item similarity, but not reason on paths. Moreover, the performance rely heavily on the quality of meta-paths, which requires domain knowledge.

Several researchers exploit programming models to infer a user preference along paths (Catherine and Cohen 2016; Chaudhari, Azaria, and Mitchell 2016). Nevertheless, these methods fail to learn representations of users and items, thus hardly generalize to unseen interactions. To solve the issues, recent studies learn the representation for each path (Hu et al. 2018; Shu et al. 2018; Wang et al. 2018c). Hu *et al.* (2018) employed CNN over the embeddings of entities to get a single representation for a path, while recurrent networks are adopted in (Shu et al. 2018). As such, these methods can combine the strengths of embedding-based and path-based approaches, achieving better performance. However, the work (Hu et al. 2018) ignores the sequential dependencies of entities and relations within a path; moreover, only entity embeddings are involved in the path modeling (Shu et al. 2018). Such limitations may hurt the reasoning ability of models. Towards this end, we propose a model to consider the sequential dependencies, as well as relation semantics, to reason a user-item interaction.

Conclusions

In this work, we exploit knowledge graph to construct paths as extra user-item connectivity, which is complementary to user-item interactions. We propose a knowledge-aware path recurrent network to generate representation for each path by composing semantics of entities and relations. By adopting LSTM on paths, we can capture the sequential dependencies of elements and reason on paths to infer user preference. Extensive experiments are performed to show the effectiveness and explainability of our model.

In future, we will extend our work in two directions. First, we attempt to mimic the propagation process of user preferences within KGs via Graph Neural Networks, since extracting qualified paths needs labor-intensive. Second, as KG links multiple domains (*e.g.,* movie and book) together with overlapped entities, we plan to adopt zero-shot learning to solve the cold start issues in the target domain.

Acknowledgement This work is supported by NExT, by the National Research Foundation Singapore under its AI Singapore Programme, Linksure Network Holding Pte Ltd and the Asia Big Data Association (Award No.: AISG-100E-2018-002). Assistance provided by eBay, Search Science Shanghai Director Hua Yang, Manager Xiaoyuan Wu, and intern Mohan Zhang was greatly appreciated.

References

[Bayer et al. 2017] Bayer, I.; He, X.; Kanagal, B.; and Rendle, S. 2017. A generic coordinate descent framework for learning from implicit feedback. In *WWW*, 1341–1350.

[Bellini et al. 2017] Bellini, V.; Anelli, V. W.; Noia, T. D.; and Sciascio, E. D. 2017. Auto-encoding user ratings via knowledge graphs in recommendation scenarios. In *DLRS@RecSys*, 60–66.

[Bordes et al. 2013] Bordes, A.; Usunier, N.; García-Durán, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*, 2787–2795.

[Cao et al. 2017] Cao, Y.; Huang, L.; Ji, H.; Chen, X.; and Li, J. 2017. Bridge text and knowledge by learning multi-prototype entity mention embedding. In *ACL*, 1623–1633.

[Catherine and Cohen 2016] Catherine, R., and Cohen, W. W. 2016. Personalized recommendations using knowledge graphs: A probabilistic logic programming approach. In *RecSys*, 325–332.

[Chaudhari, Azaria, and Mitchell 2016] Chaudhari, S.; Azaria, A.; and Mitchell, T. M. 2016. An entity graph based recommender system. In *RecSys*.

[Chen et al. 2017] Chen, J.; Zhang, H.; He, X.; Nie, L.; Liu, W.; and Chua, T.-S. 2017. Attentive collaborative filtering: Multimedia recommendation with item- and component-level attention. In *SIGIR*, 335–344.

[Gao et al. 2018] Gao, L.; Yang, H.; Wu, J.; Zhou, C.; Lu, W.; and Hu, Y. 2018. Recommendation with multi-source heterogeneous information. In *IJCAI*, 3378–3384.

[He and Chua 2017] He, X., and Chua, T. 2017. Neural factorization machines for sparse predictive analytics. In *SIGIR*, 355–364.

[He et al. 2017] He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; and Chua, T. 2017. Neural collaborative filtering. In *WWW*, 173–182.

[He et al. 2018] He, X.; He, Z.; Du, X.; and Chua, T. 2018. Adversarial personalized ranking for recommendation. In *SIGIR*, 355–364.

[Heitmann and Hayes 2010] Heitmann, B., and Hayes, C. 2010. Using linked data to build open, collaborative recommender systems. In *AAAI*.

[Hu et al. 2018] Hu, B.; Shi, C.; Zhao, W. X.; and Yu, P. S. 2018. Leveraging meta-path based context for top- N recommendation with A neural co-attention model. In *SIGKDD*, 1531–1540.

[Huang et al. 2018] Huang, J.; Zhao, W. X.; Dou, H.; Wen, J.; and Chang, E. Y. 2018. Improving sequential recommendation with knowledge-enhanced memory networks. In *SIGIR*, 505–514.

[Lin et al. 2015] Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; and Zhu, X. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, 2181–2187.

[McCallum et al. 2017] McCallum, A.; Neelakantan, A.; Das, R.; and Belanger, D. 2017. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *EACL*, 132–141.

[Neelakantan, Roth, and McCallum 2015] Neelakantan, A.; Roth, B.; and McCallum, A. 2015. Compositional vector space models for knowledge base completion. In *ACL*, 156–166.

[Nickel, Tresp, and Kriegel 2011] Nickel, M.; Tresp, V.; and Kriegel, H. 2011. A three-way model for collective learning on multi-relational data. In *ICML*, 809–816.

[Rendle et al. 2009] Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme, L. 2009. BPR: bayesian personalized ranking from implicit feedback. In *UAI*, 452–461.

[Shi et al. 2015] Shi, C.; Zhang, Z.; Luo, P.; Yu, P. S.; Yue, Y.; and Wu, B. 2015. Semantic path based personalized recommendation on weighted heterogeneous information networks. In *CIKM*, 453–462.

[Shu et al. 2018] Shu, Z.; Yang, J.; Zhang, J.; Bozzon, A.; Huang, L.-K.; and Xu, C. 2018. Recurrent knowledge graph embedding for effective recommendation. In *RecSys*.

[Sun and Han 2012] Sun, Y., and Han, J. 2012. Mining heterogeneous information networks: a structural analysis approach. *SIGKDD* 14(2):20–28.

[Sun et al. 2011] Sun, Y.; Han, J.; Yan, X.; Yu, P. S.; and Wu, T. 2011. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *PVLDB* 4(11):992–1003.

[Wang et al. 2017] Wang, X.; He, X.; Nie, L.; and Chua, T. 2017. Item silk road: Recommending items from information domains to social users. In *SIGIR*, 185–194.

[Wang et al. 2018a] Wang, H.; Zhang, F.; Wang, J.; Zhao, M.; Li, W.; Xie, X.; and Guo, M. 2018a. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *CIKM*, 417–426.

[Wang et al. 2018b] Wang, H.; Zhang, F.; Xie, X.; and Guo, M. 2018b. DKN: deep knowledge-aware network for news recommendation. In *WWW*, 1835–1844.

[Wang et al. 2018c] Wang, X.; He, X.; Feng, F.; Nie, L.; and Chua, T. 2018c. TEM: tree-enhanced embedding model for explainable recommendation. In *WWW*, 1543–1552.

[Yu et al. 2013] Yu, X.; Ren, X.; Gu, Q.; Sun, Y.; and Han, J. 2013. Collaborative filtering with entity similarity regularization in heterogeneous information networks. *IJCAI* 27.

[Yu et al. 2014] Yu, X.; Ren, X.; Sun, Y.; Gu, Q.; Sturt, B.; Khandelwal, U.; Norick, B.; and Han, J. 2014. Personalized entity recommendation: a heterogeneous information network approach. In *WSDM*, 283–292.

[Zhang et al. 2016] Zhang, F.; Yuan, N. J.; Lian, D.; Xie, X.; and Ma, W. 2016. Collaborative knowledge base embedding for recommender systems. In *SIGKDD*, 353–362.

[Zhao et al. 2017] Zhao, H.; Yao, Q.; Li, J.; Song, Y.; and Lee, D. L. 2017. Meta-graph based recommendation fusion over heterogeneous information networks. In *SIGKDD*, 635–644.