Recurrent Knowledge Graph Embedding for Effective Recommendation

Zhu Sun[1], Jie Yang[2], Jie Zhang[1], Alessandro Bozzon[3], Long-Kai Huang[1], Chi Xu[4*]
[1]Nanyang Technological University, Singapore
[2]eXscale Infolab, University of Fribourg, Fribourg, Switzerland
[3]Delft University of Technology, Delft, The Netherlands
[4]Singapore Institute of Manufacturing Technology, Singapore
{sunzhu,zhangj,lhuang018}@ntu.edu.sg,jie@exascale.info,a.bozzon@tudelft.nl,cxu@simtech.a-star.edu.sg

ABSTRACT

Knowledge graphs (KGs) have proven to be effective to improve recommendation. Existing methods mainly rely on hand-engineered features from KGs (e.g., meta paths), which requires domain knowledge. This paper presents RKGE, a KG embedding approach that automatically learns semantic representations of both entities and paths between entities for characterizing user preferences towards items. Specifically, RKGE employs a novel recurrent network architecture that contains a batch of recurrent networks to model the semantics of paths linking a same entity pair, which are seamlessly fused into recommendation. It further employs a pooling operator to discriminate the saliency of different paths in characterizing user preferences towards items. Extensive validation on real-world datasets shows the superiority of RKGE against state-of-the-art methods. Furthermore, we show that RKGE provides meaningful explanations for recommendation results.

CCS CONCEPTS

• **Information systems** → **Collaborative filtering**;

KEYWORDS

Knowledge Graph; Recurrent Neural Network; Semantic Representation; Attention Mechanism

1 INTRODUCTION

Knowledge graphs (KGs) as an auxiliary data source have recently attracted a considerable amount of interest to enhance recommendation. They connect various types of information related to items (e.g., genre, director, actor of a movie) in a unified global space, which helps to develop insights on recommendation problems that are difficult to uncover with user-item interaction data only. State-of-the-art methods [15, 28, 37, 42, 45] mainly extend the latent factor model (LFM) [29] with entity similarity derived from paths (e.g., meta paths [33]) in a KG, based on the intuition that paths connecting two entities represent entity relations of different semantics. Such an intuition facilitates the inference of user preferences based on item similarity for generating effective recommendations. Meta path based methods, however, heavily rely on handcrafted features to represent path semantics, which further relies on domain knowledge. More importantly, manually designed features are often incomplete to cover all possible entity relations, thus hindering the improvement of the recommendation quality.

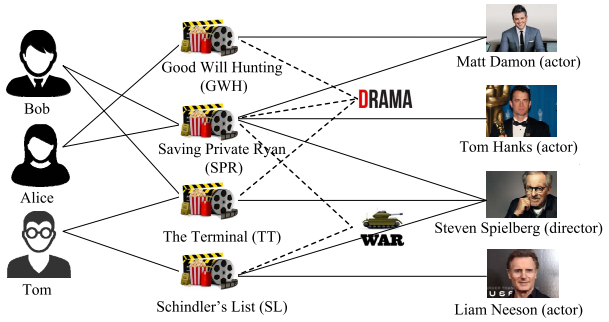

Figure 1: A KG in the movie domain, which contains users, movies, actors, directors and genres as entities; rating, categorizing, acting, and directing as the entity relations.

The popularity of representation learning (RL) recently prompted a seminal work [44] that exploits KG embedding to capture entity semantics for recommendation. In contrast to meta path based methods relying on handcrafted features, KG embedding based methods automatically learn the embeddings of entities in KGs by using the one level ego-network of entities with their properties. As a result, they have achieved higher performance than meta path based methods. A major limitation of these methods is, however, the disregard of semantic relations of entities that are connected by paths, which has been extensively studied in meta path based methods. We therefore seek for a new data-driven method that does not depend on handcrafted features (e.g., meta paths), yet can capture the semantics of *both* entities and paths encoded in KGs for recommendation. Figure 1 illustrates the need for modeling path semantics for recommendation as well as the challenges.

Running Example. Consider a KG based movie recommender system, where Bob's preference over SPR[1] can be inferred by: 1) Bob $\xrightarrow{rate}$ TT $\xrightarrow{categorized\ by}$ Drama $\xrightarrow{categorize}$ SPR; 2) Bob $\xrightarrow{rate}$ TT

*Both the first two authors contributed equally to the paper.

[1]For all the movies, we adopt the abbreviation for short.

RecSys '18, October 2–7, 2018, Vancouver, BC, Canada

ACM ISBN 978-1-4503-5901-6/18/10…$15.00
https://doi.org/10.1145/3240323.3240361

$\xrightarrow{directed\ by}$ Steven Spielberg $\xrightarrow{direct}$ SPR. These paths capture semantic relations of 1) belonging to the same genre, or 2) being directed by the same director for the movies that Bob has watched. Hence, we may infer that Bob prefers either movies belonging to the genre of Drama or those directed by Steven Spielberg. Based on these insights, we can recommend GWH (belongs to Drama) or SL (directed by Steven Spielberg) to Bob. This example highlights that different paths connecting a same entity pair often carry relations of different semantics. Typically, they are of different importance in characterizing user tastes over items, i.e., certain paths can better describe user preferences than the others. In the example, Bob's preference over SPR may be driven more by his interest in the genre than by his favor for the director. To fully exploit paths in KGs for recommendation, it requires to capture not only the semantics of different paths but also their distinctive saliency in describing user preferences towards items.

To this end, we consider to adopt recurrent networks [5, 39] to learn the semantics of entity relations. Recurrent networks are capable of modeling sequences with various lengths, making it particularly suitable for modeling paths – i.e., sequences of different numbers of entities – in KGs. Most importantly, recurrent networks can not only model the semantics of entities (via an embedding layer [34]), but also those of entity relations by encoding the entire path, thus providing a unified approach to learn the representations of both entities and entity relations. Given different descriptive power of paths in characterizing user preferences, it is however non-trivial to model all relevant paths in KGs by the standard recurrent network architecture.

To exploit KGs for recommendation as well as to address the above challenge, we propose a unified recurrent knowledge graph embedding framework RKGE. RKGE first automatically mines all qualified paths between entity pairs from the KG, which are then encoded via a batch of recurrent networks, with each path modeled by a single recurrent network. The recurrent networks in the batch share common parameters to avoid over-fitting. RKGE is thus flexible in capturing different numbers of paths with various lengths that connect entity pairs. It then employs a pooling operation to discriminate the importance of different paths for characterizing user preferences towards items. Finally, a recommendation layer is seamlessly integrated with the network such that RKGE can be trained in an end-to-end manner. To the best of our knowledge, this is the first work that adapts recurrent networks to capture the semantics of both entities and paths encoded in KGs for effective recommendation. Extensive experiments on real-world datasets show that RKGE consistently outperforms the state-of-the-art with a lift of 17.84% in Precision and 11.82% in MRR on average.

2 RELATED WORK

This section provides an overview of state-of-the-art methods that utilize KG for better recommendation. They are generally classified into three types, namely graph based methods, meta path based methods and KG embedding based ones.

Graph based Methods. A line of research focuses on making use of KGs by designing graph based methods. Early work [7] proposes a method by applying the spreading activation technique [22] on KGs. This results in a model that provides lower rating estimation error and higher coverage for recommendation compared to those collaborative filtering methods using only user-item interactions. Later, Pham et al. [20, 21] propose HeteRS to solve recommendation problems in event-based social networks. They transform the recommendation problem into a node proximity calculation problem and employ Markov chain to solve it. After that, Catherine and Cohen [2] investigate a recommendation approach by adopting logic reasoning on KGs to infer user preferences. Recently, Chaudhari et al. [3] introduce RERA, a recommender system that adopts Personalized Page Rank to utilize KGs for better recommendation.

All these graph based methods are attributed to the underlying technique of random walk [6], which however can be easily biased to popular and centered entities in KGs. More importantly, they only consider the topological structure of KGs without considering to model the semantics of entities and entity relations encoded in the KG, thus failing to fully exploit KGs for recommendation.

Meta Path based Methods. Another set of methods utilizes KGs by designing meta paths, which predefine the specific format and length of the paths to capture different semantics carried by KGs.

Several studies leverage the relations of items connected by meta paths to boost recommendation quality. For instance, Yu et al. [41] devise HeteMF – a matrix factorization [16] framework with meta-path-based entity similarity. It decomposes the user-item rating matrix, meanwhile adopting graph regularization [31] to constrain the distance of latent vectors of similar items that are connected by meta paths. Later, they propose HeteRec [43] to learn user preference diffusion to the unrated items that are connected with her rated items via different meta paths in KGs. HeteRec is designed for implicit feedback and estimated by the Bayesian ranking optimization [24], and is further extended to incorporate personalization via clustering users based on their interests by Yu et al. [42].

Other work models the relations of user-user or user-item via meta paths. For example, Luo et al. [15] investigate a social network-based recommendation algorithm HeteCF to model the relations of user-item, user-user and item-item by meta-path based similarity. In order to accurately capture semantic relations among users, Shi et al. [28] propose the SemRec model and introduce the concept of weighted meta path, which aims to depict the path semantics by distinguishing subtle differences among link attribute values. Later, the same authors design a matrix factorization based dual regularization framework SimMF [27]. They design regularization terms for both users and items with the help of meta path based similarity. Similarly, Wang et al. [37] and Zheng et al. [45] also devise matrix factorization approaches by regularizing user-user relations with the computed meta path based similarity.

The success of these methods heavily relies on the quality and quantity of the handcrafted meta paths, which additionally requires on domain knowledge. This largely limits the capability of these methods for generating high-quality recommendation.

KG Embedding based Methods. The most recent state-of-the-art algorithm is collaborative knowledge graph embedding (CKE) proposed by Zhang et al. [44]. CKE learns better item latent representations by capturing entity semantics from KGs via TransR [12], showing the superior performance over other existing methods that exploit KGs for recommendation. However, it ignores the semantics of the relations between paired entities represented by

paths, thus failing to fully capture KG semantics for recommendation. In contrast, our RKGE models the semantics of all the paths between entities in a fully automatic fashion using a batch of recurrent networks, meanwhile it learns the respective path saliency with a pooling operation. By doing so, RKGE advances the current state-of-the-art embedding based recommendation methods by fully exploiting entity relations in KGs.

Our method is related to graph embedding based methods, such as node2Vec [8], LINE [35], and DeepWalk [19], as proposed for network analysis (e.g., node classification and link prediction). However, they are not suitable for our case, as they all aim to learn entity representations in a homogeneous network, whereas KGs studied here are heterogeneous. We further note that graph embedding based methods have also been applied to other domains, e.g, semantic search [13, 14]. However, we mainly focus on methods that fall into the area of recommendation in this study.

3 RECURRENT KNOWLEDGE GRAPH EMBEDDING

Given user-item interaction data, our goal is to exploit the heterogeneous information encoded in the KG to help learn high-quality representations of both users and items, which are then used to generate better recommendations. The extracted representations are expected to fully capture the semantic meanings of entities and entity relations encoded in the KG. To achieve this goal, we propose the recurrent knowledge graph embedding approach (RKGE). The overall framework is illustrated in Figure 2. RKGE first automatically mines semantic paths between entity pairs, then employs a novel recurrent network architecture to encode different paths via a batch of recurrent networks. It further determines different path saliency through a pooling operation, which in the end, is seamlessly integrated with recommendation generation.

Notations. Table 1 summarizes the notations used throughout this paper. We denote the user set as $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$ and the item set as $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$, and use $\mathbf{R} \in \mathbb{R}^{m \times n}$ to denote historical user-item interactions, with $r_{ij} = 1$ indicating that u_i prefers v_j and 0 otherwise. We use *entity* as a generic term to refer to all relevant objects (e.g., user, item, genre, actor) that can be mapped into a KG, denoted as $\mathcal{G}$. The definition of KG is given as below.

Definition 1. Knowledge Graph. Let $\mathcal{E} = \{e_1, e_2, \ldots, e_k\}$ and $\mathcal{R} = \{r_1, r_2, \ldots, r_g\}$ denote the sets of entities and entity relations, respectively. KG is defined as a directed graph $\mathcal{G} = (\mathcal{E}, \mathcal{L})$ with an entity type mapping function $\phi : \mathcal{E} \rightarrow \mathcal{A}$ and a link type mapping function $\psi : \mathcal{L} \rightarrow \mathcal{R}$. Each entity $e \in \mathcal{E}$ belongs to an entity type $\phi(e) \in \mathcal{A}$, and each link $l \in \mathcal{L}$ belongs to a link type (relation) $\psi(l) \in \mathcal{R}$. Finally, we use $\mathcal{P}(e_i, e_j) = \{p_1, p_2, \ldots p_s\}$ to represent the connected paths between entities e_i and e_j.

The KG investigated in this study can be considered as a heterogeneous information network, as there are more than one types of entities and entity relations included, i.e., $|\mathcal{A}| > 1$ and/or $|\mathcal{R}| > 1$. Figure 1 provides a toy example of the KG in the movie domain, where entities include user, movie and the corresponding attributes (e.g., genre, actor and director), and links describe the relations between entities (e.g., "rating" behavior and "acting" behavior).

Table 1: Notations

Notations	Descriptions
$\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$	User set
$\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$	Item set
$\mathbf{R} \in \mathbb{R}^{m \times n}$	User-item interaction matrix
$r_{ij}, \tilde{r}_{ij}$	Observed and estimated ratings
$\mathcal{E} = \{e_1, e_2, \ldots, e_k\}$	Entity set
$\mathcal{R} = \{r_1, r_2, \ldots, r_g\}$	Entity relation set
$\mathcal{G} = (\mathcal{E}, \mathcal{L})$	Knowledge graphs
$\mathcal{P}(e_i, e_j) = \{p_1, p_2, \ldots p_s\}$	Paths between entity pair (e_i, e_j)
$p_l = e_0 \rightarrow e_1 \cdots \rightarrow e_T$	Path p_l between entity pair
$\mathbf{p}_l = \{\mathbf{p}_{l0}, \mathbf{p}_{l1}, \cdots, \mathbf{p}_{lT}\}$	Embedding of path p_l
a_{lt}	Attention gate at current step
$\mathbf{h}_{lt}$	Current hidden state
$\mathbf{h}'_{lt}$	Current candidate hidden state
$\mathbf{h}$	Aggregated hidden state
$\mathbf{W}, \mathbf{H}$	Linear transformation parameters
σ	Sigmoid activation function
$\mathcal{J}$	Objective function

3.1 Semantic Path Mining

To fully exploit entity relations in KGs, we first mine paths with different semantics between entities, which are then seamlessly fused into the recurrent network batch for effective recommendation. Due to the large volume and complexity of KGs, there are a large number of paths connecting entity pairs that may contain different entity types and relation types in different orders and with various lengths. To increase model efficiency, we thus devise two strategies to help select salient paths:

Strategy 1. *We only consider user-to-item paths $\mathcal{P}(u_i, v_j)$ that connect u_i with all her rated items, i.e., $\{v_j | r_{ij} > 0\}$. These paths are most helpful for recommendation given our goal to recommend items to users. Moreover, they further include those relevant item-to-item and user-to-user paths as subsequences of user-to-item paths.*

Strategy 2. *We enumerate paths with a length constraint, i.e., only paths with length less than a threshold are employed. As pointed out by Sun et al. [33], paths with relatively short length are good enough to model entity relations, whereas longer paths may bring in remote neighbors and lose semantic meanings, thus introducing much noise.*

We will also investigate how the lengths of paths can affect recommendation performance in our new context of knowledge graph embedding based approach and show in our experiment that similar results also hold. Guided by the two strategies, RKGE mines qualified paths with different semantics that connect entity pairs (i.e., user-item) in an automatic fashion, instead of manually designed and extracted features (e.g., meta paths) from the KG. These paths will be further processed by the recurrent network batch to automatically learn their semantic representations for recommendation, as will be introduced next.

3.2 Recurrent Network Batch

By regarding the directed path between user-item pair as a sequence, where elements in the sequence are the entities in the path, we naturally consider to adopt recurrent networks to encode the path.

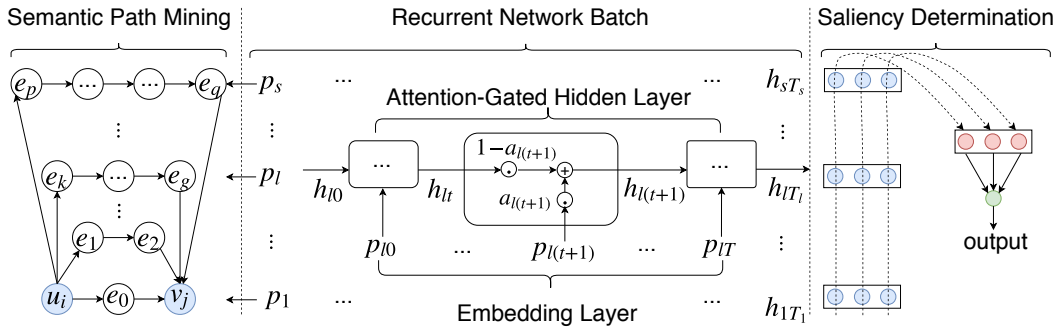

Figure 2: The overall framework of RKGE, which describes the case of a user-item pair.

This is mainly attributed to their capability in modeling sequences with various lengths and in capturing the semantics of both entities and the entire path between entity pair. Given that multiple paths with various lengths may connect entity pair, we devise a novel network architecture to capture all possible relations, which comprises a batch of recurrent networks, with every single recurrent network learning the semantic representation of an individual path. As the number of paths between different entity pairs is dynamic, the number of recurrent networks in the batch varies in accordance with that of connected paths between the entity pair. Besides, all the recurrent networks in the batch share the same parameters to further avoid over-fitting.

Assume s paths of different lengths connect an entity pair (u_i, v_j), i.e., $\mathcal{P}(u_i, v_j) = \{p_1, p_2, \cdots, p_s\}$. Note that s is dynamic, as different entity pairs may be connected with different number of paths. For any path p_l with length T in the format of $p_l = e_0 \xrightarrow{r_1} e_1 \xrightarrow{r_2} e_2 \cdots \xrightarrow{r_T} e_T$ with $e_0 = u_i, e_T = v_j$, the recurrent network encodes the path by learning a semantic representation for each entity and a single representation for the entire path. In RKGE, these goals are achieved by two network layers, namely the embedding layer and the attention-gated hidden layer, as illustrated in Figure 2.

Embedding Layer. For each entity e_t in p_l, the embedding layer learns a distributed representation $\mathbf{p}_{lt}$ that maps e_t into a low dimensional vector, with each element of the vector representing the affinity of this entity to a latent topic, thus capturing the semantic meaning of the entity. This results in a representation of path p_l as $\mathbf{p}_l = \{\mathbf{p}_{l0}, \mathbf{p}_{l1}, \mathbf{p}_{l2}, \cdots, \mathbf{p}_{lT}\}$, where each element denotes the representation (embedding) of the corresponding entity in p_l. This new representation will then be fed as input to the hidden layer to learn a single representation that encodes the entire path.

Attention-Gated Hidden Layer. To learn the path representation, the hidden layer considers both the embeddings of entities in the path and the order of these entities. It takes a flow-based approach to encode the sequence from the beginning entity of the path e_0 to the ending entity e_T: in each step $t-1$, it learns a hidden state $\mathbf{h}_{l(t-1)}$ that encodes the subsequence from e_0 to e_{t-1}, which is then used as input together with the embedding of e_t (i.e., $\mathbf{p}_{lt}$) to learn the hidden state of the next time step, i.e., $\mathbf{h}_{lt}$. The final state $\mathbf{h}_{lT}$ will encode the entire path, thus is considered as the representation of the whole path.

We propose to use an attention gate to better control information flows through path p_l, which has proven to be more effective than plain recurrent neural networks [17, 18, 40]. We denote the attention gate at step t by a_{lt}, which is a scalar value between $[0, 1]$. The hidden-state at time t is modeled as:

$$\mathbf{h}_{lt} = (1 - a_{lt}) \cdot \mathbf{h}_{l(t-1)} + a_{lt} \cdot \mathbf{h}'_{lt} \tag{1}$$

where the attention gate a_{lt} balances the contributions of the input of the previous hidden-state $\mathbf{h}_{l(t-1)}$ and the current candidate hidden-state $\mathbf{h}'_{lt}$. The current *candidate* hidden-state is further given by fully incorporating the input at the current time step:

$$\mathbf{h}'_{lt} = \sigma\left(\mathbf{W} \cdot \mathbf{h}_{l(t-1)} + \mathbf{H} \cdot \mathbf{p}_{lt} + b\right) \tag{2}$$

where $\mathbf{W}, \mathbf{H}$ are the linear transformation parameters for the previous and current steps, respectively; b is the bias term; σ is the sigmoid activation function.

Finally, the attention gate is inferred by using a bi-directional recurrent networks [26] to maximize the exploration of the input sequence. We model the attention gate based on both the input observation at the current time step and the information from neighboring observation in both directions, formulated by

$$a_{lt} = \sigma\left(\mathbf{M}^\top \cdot (\overrightarrow{\mathbf{h}_{lt}}; \overleftarrow{\mathbf{h}_{lt}}) + b'\right) \tag{3}$$

where σ is the sigmoid activation function to control the range of attention gate into range $[0, 1]$; $\mathbf{M}$ is the weight vector and b' is the bias term of the attention layer; $(;)$ denotes the concatenation among vectors; $\overrightarrow{\mathbf{h}_{lt}}$ and $\overleftarrow{\mathbf{h}_{lt}}$ are the hidden representations of a bi-directional recurrent network model [26], performed as the summary of context information around time step t, given by,

$$\begin{aligned} \overrightarrow{\mathbf{h}}_{lt} &= \sigma\left(\overrightarrow{\mathbf{W}} \cdot \overrightarrow{\mathbf{p}}_{lt} + \overrightarrow{\mathbf{H}} \cdot \overrightarrow{\mathbf{h}}_{l(t-1)} + \overrightarrow{b}\right) \\ \overleftarrow{\mathbf{h}}_{lt} &= \sigma\left(\overleftarrow{\mathbf{W}} \cdot \overleftarrow{\mathbf{p}}_{lt} + \overleftarrow{\mathbf{H}} \cdot \overleftarrow{\mathbf{h}}_{l(t+1)} + \overleftarrow{b}\right) \end{aligned} \tag{4}$$

Thus, $\overrightarrow{\mathbf{h}}_{lt}$ summarizes the path from the beginning to step t, while $\overleftarrow{\mathbf{h}}_{lt}$ summarizes the path from the end to step t.

Overall, by incorporating each qualified path (with a total number of s) between u_i and v_j into the corresponding attention-gated recurrent network simultaneously, the result is a recurrent network batch, with each attention-gated recurrent network encoding a single path. To avoid over-fitting, all the attention-gated recurrent networks in the batch share the same parameters. Finally, we obtain the hidden representations of all paths, i.e., the representations of the entity relations of u_i and v_j. The different importance of these

hidden states on modeling entity relations is then distinguished by a pooling operation, as we will elaborate next.

3.3 Saliency Determination

As there are s paths linking u_i and v_j, different paths may play different roles in modeling the relations between them. For example, previous work has shown that shorter paths may have more impacts than longer ones, as shorter paths often indicate a stronger connectivity with clearer semantics. Hence, we design a data-driven method via pooling operations [11] to help distinguish the path importance. Attention mechanisms [40] also seem to be one possible solution to address this issue. However, it generally aims to identify the importance of each element in a single sequence, while our goal is to decide the saliency of each path (i.e., sequence) between an entity pair. We consider to use pooling operations, which are designed to focus on the most important "features" of different vectors, thus are more suitable for our purpose.

For the s paths in $\mathcal{P}(u_i, v_j)$, their last hidden states learnt by the recurrent network batch are $\mathbf{h}_{1T_1}, \mathbf{h}_{2T_2}, \cdots, \mathbf{h}_{sT_s}$, where T_s is the last step of p_s as well as the length of p_s. Based on this, we add a *max pooling* layer to get the most salient feature across all the paths. This results in an aggregated hidden state $\mathbf{h}$:

$$\mathbf{h}[j] = \max_{1 \le i \le s} \mathbf{h}_{iT_i}[j] \tag{5}$$

where $\mathbf{h}[j]$ is the value of the j^{th} dimension of $\mathbf{h}$. Furthermore, to avoid $\mathbf{h}$ being dominated by a certain $\mathbf{h}_{iT_i}$, e.g., a single path in $\mathcal{P}(u_i, v_j)$, we also perform an *average pooling* operation [1] towards the last hidden states of all the paths. Their respective performance will be evaluated in the experiment section later.

Through the pooling operation, we get a final hidden state of all the paths between u_i and v_j, i.e., the aggregated effects of paths on the relation of u_i and v_j. We then adopt a fully-connected layer after the pooling layer to further quantify the relation (proximity) of u_i and v_j, i.e., $\tilde{r}_{ij}$, given by:

$$\tilde{r}_{ij} = f(\mathbf{h}) = \sigma(\mathbf{W}_r \cdot \mathbf{h} + b_r) \tag{6}$$

where $\mathbf{W}_r$ is regression coefficient and b_r is the bias term. We adopt a sigmoid function $\sigma(\cdot)$ to control the range of $f(\mathbf{h})$ into $[0, 1]$.

Once model training is finished, better representations of u_i and v_j can be achieved by encoding the connected paths between them via RKGE. Following [44], during the test process, we calculate the proximity score of user u_i and items v_k via the inner product of their corresponding embeddings [25], i.e., $s(u_i, v_k) = \langle \mathbf{u}_i, \mathbf{v}_k \rangle$. Inner product is used as it is more efficient than predicting a user's rating via the network, which requires a time-consuming feed-forward pass through the whole network. Finally, we rank the items based on the proximity score, and recommend the top-K items with the highest score to u_i.

3.4 Model Optimization

Model Learning. Given the training data $\mathcal{D}_{\text{train}}$, which contains instances in the form of $\big(u_i, v_j, r_{ij}, \mathcal{P}(u_i, v_j)\big)$, RKGE learns the involved parameters by minimizing the following loss function:

$$\mathcal{J} = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum\nolimits_{r_{ij} \in \mathcal{D}_{\text{train}}} BCELoss(\tilde{r}_{ij}, r_{ij}) \tag{7}$$

Algorithm 1: RKGE Optimization

Input: rating matrix $\mathbf{R}$, knowledge graph $\mathcal{G}$, in_dim, hidden_dim, max_path_length, γ, Iter

1 Initialize the embeddings of $e \in \mathcal{G}$ with small values;
// Semantic Path Mining
2 Built the graph $\mathcal{G}$ with Python Networkx;
3 **foreach** $u_i \in \mathcal{U}$ **do**
4 Based on $\mathbf{R}$, get positive pairs (u_i, v_j);
5 Randomly sample to generate negative pairs (u_i, v_k);
6 $(u, v) \leftarrow (u_i, v_j) \cup (u_i, v_k)$;
7 **foreach** (u, v) *pair* **do**
8 Mine connected paths $\mathcal{P}(u, v)$;
// Recurrent Network Architecture
9 **for** $t = 1; t \le Iter; t++$ **do**
10 **foreach** (u, v) *pair* **do**
 // Recurrent Network Batch
11 **for** $p_l \in \mathcal{P}(u, v)$ **do**
12 $\mathbf{h}_{lT_l} \leftarrow$ based on Equ. (1-4);
13 Combine$(\mathbf{h}) \leftarrow \mathbf{h}_{lT_l}$;
 // Saliency Determination
14 $\mathbf{h} \leftarrow pool\big(\text{Combine }(\mathbf{h})\big)$ based on Equ. (5);
15 Calculate $\tilde{r}_{ij}$ based on Equ. (6);
16 Update parameters by back propagation through time;

where BCELoss$(\cdot)$ is the Binary Cross Entropy between the observed and estimated ratings. Thus, we address the recommendation problem as a binary classification problem, the effectiveness of which has proven by [9]. As Equation 7 and all modules of RKGE are analytically differentiable, it can be readily trained in an end-to-end manner. Parameters are updated by the back propagation through time (BPTT) algorithm [38] in the recurrent layers and by normal back-propagation in other parts. We randomly sample unrated items for each user as negative instances, the number of which is the same with her rated items. The paths connecting users and their negative instances are also exploited to help balance model learning. The detailed optimization process is described by Algorithm 1, which is mainly composed of two modules: the semantic path mining (lines 2-8) and the recurrent network architecture (lines 9-16) module, which includes the recurrent network batch (lines 11-13) and saliency determination (lines 14-15).

Model Scalability. We design a two-level parallel training mechanism for RKGE to improve model scalability. In each iteration of model training, the connection paths of all user-to-item pairs for individual users are fed into RKGE in parallel. Meanwhile, parameters related to the multiple paths between each user-to-item pair are simultaneously updated. As a result, the training time for RKGE on the two evaluation datasets, i.e., Movielens-1M and Yelp, (see Section 4.1) is around three hours and one hour, respectively. To summarize, RKGE is scalable to large datasets and KGs.

4 EXPERIMENTS AND ANALYSIS

4.1 Experimental Setup

Datasets. To demonstrate the effectiveness of our proposed recommendation framework, we adopt two real-world datasets from

Table 2: Dataset statistics.

Datasets		IM-1M	Yelp
User-item interaction	#Users	6,040	37,940
	#Items	3,382	11,516
	#Ratings	756,684	229,178
	Data Density	3.704%	0.052%
Knowledge graph	#Entities	18,920	46,606
	#Entity Types	11	7
	#Links	800,261	302,937
	#Link Types	10	6
	Graph Density	0.447%	0.028%

different domains (movie and local business) for empirical study. The first dataset is IM-1M, which is built by combing MovieLens 1M[2] and the corresponding IMDB[3] datasets. MovieLens 1M is a personalized movie rating dataset, which consists of 1M ratings (ranging from 1 to 5) with 6,040 users and 3,706 movies; IMDB contains movie auxiliary information, such as genre, actor, director, etc. The two datasets are linked by the titles and release dates of movies. After mapping the two datasets, we have 6,040 users and 3,382 movies, and 756,684 ratings in the final dataset. The second dataset is Yelp, which is the Yelp Challenge Dataset[4] released by Yelp[5] and is available now at Kaggle[6]. This dataset contains user check-ins to local business, together with user reviews and local business information network (e.g., category, location). Yelp is much sparser than IM-1M, thus the performances of all the methods are expected to decline accordingly on Yelp.

We process the two datasets in accordance with literature [2, 42] as follows: if a user provides a rating towards a movie, or wrote a review for a business, we set the feedback as 1, otherwise it would be set to 0. We split the feedback of IM-1M in the order of their timestamps, and use the older 80% of feedback as training data and the more recent 20% as test data. With Yelp we utilize the original training and test sets in the published version. The overall statistics of the two datasets are summarized in Table 2.

Comparison Methods. To validate the effectiveness of our proposed framework, we compare with several algorithms: 1) **MostPop**: recommends the most popular items to all users without personalization; 2) **BPRMF** [24]: Bayesian personalized ranking model based on matrix factorization (MF) [16]; 3)**NCF** [9]: neural network based collaborative filtering recommendation method; 4) **LIBFM** [23]: factorization machine is a classic feature based latent factor model, to which we feed item attributes in the KG as raw features; 5) **HeteRS** [20, 21]: the graph based recommendation method integrating KGs via Markov chain; 6) **HeteRec** [42]: latent factor model fusing meta path for personalized recommendation; 7) **GraphLF** [2]: the graph based approach using Personalized PageRank to infer user preferences via logic reasoning; 8) **CKE** [44]: the recently proposed state-of-the-art collaborative KG embedding based method that learns better item latent representations with the help of KGs.

[2]http://grouplens.org/datasets/movielens/
[3]http://www.imdb.com/
[4]https://www.yelp.com/dataset_challenge
[5]http://www.yelp.com/
[6]https://www.kaggle.com/c/yelp-recsys-2013/data

For fair comparison, we remove the textual and visual embedding parts from the original CKE, due to the unavailable of such auxiliary information in the evaluation datasets. Besides, we take LIBFM as the representative feature based latent factor model, as it generally outperforms other counterparts, such as SVDFeature [4], collective matrix factorization (CMF) [30] and tensor factorization (TF) [10]. We do not compare with other meta path based methods (e.g., HeteCF [15], SimMF [27]), as they are generally outperformed by either HeteRec or GraphLF. Note that we do not compare with RRN [39], as it aims to study the impacts of temporal dynamics, which is beyond our research scope.

Evaluation Metrics. In alignment with literature [2, 42, 43], we adopt Precision at $N = \{1, 5, 10\}$, i.e., $Prec@N$, and the top-N Mean Reciprocal Rank (MRR) [36], as evaluation metrics.

$$MRR_N = \frac{1}{m}\sum_{i=1}^{m}\left(\sum_{v_j \in test(u_i)} \frac{1}{rank(u_i, v_j)}\right) \tag{8}$$

where we set $N = 10$; v_j is the correctly recommended item in the top-N list; $test(u_i)$ is the set of items in the test data for u_i; $rank(u_i, v_j)$ is the position of v_j in u_i's recommendation list.

Parameter Settings. The optimal parameter settings for all the methods are empirically found out. We apply a grid search in $\{10, 20, 50, 100, 200\}$ to find out the best settings for the dimension of latent factor d. A grid search in $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ is applied for the learning rate and regularization coefficient (including the 1/2-way regularization of LIBFM); for HeteRS, $\beta = 10^4$. For HeteRec and CKE, a grid search in $\{5, 10, 20, 50, 100, 200\}$ is applied to find out the best settings for the number of negative samples; the rest parameters are set as suggested by the original papers. We set the parameters of NCF as suggested by [9]. For RKGE, the number of hidden units is set to 32 on IM-1M and Yelp, which is selected from $\{8, 16, 32, 64, 128\}$ based on a held-out validation set. To avoid potential over-fitting, the dropout value is validated from the option set $\{0.00, 0.25, 0.50\}$ [32].

4.2 Results of RKGE

Impacts of Path Lengths. Paths with various lengths capture different semantics, which help to infer user preferences from different angles and allow to generate different recommendations. Our hypothesis is that paths with relatively short lengths are more beneficial for modeling entity relation as they carry clearer and interpretable semantic meanings. This has been verified in meta path based method [33]. Here we study if the same result holds by KG embedding based approach. To empirically investigate the impact of path length on recommendation accuracy, we incorporate paths with different lengths, i.e., $L = \{3, 5, 7\}$ into the proposed RKGE model. As RKGE aims at modeling paths of user-to-item pairs, and these paths are of odd lengths since items can only be indirectly linked via their attribute entities. Figure 3 depicts the results on the two datasets. From the results, we observe that as path lengths increase, the performance (including Pre@1, 5, 10 and MRR) of RKGE decreases gradually on both datasets. This verifies our intuition and confirms previous findings in the new context of KG embedding based approach.

Impacts of Pooling Operations. To determine the saliency of different paths between two entities, pooling operations are adopted

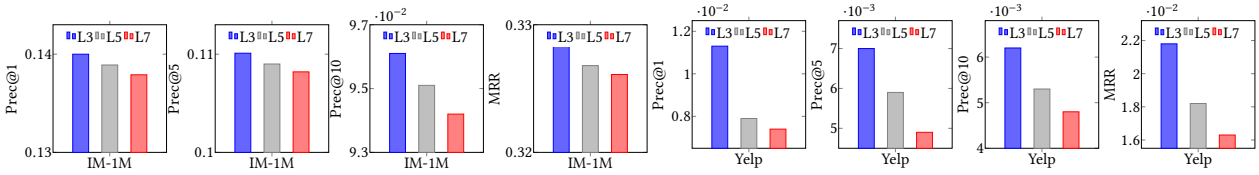

Figure 3: The impacts of different path lengths for RKGE on IM-1M and Yelp datasets.

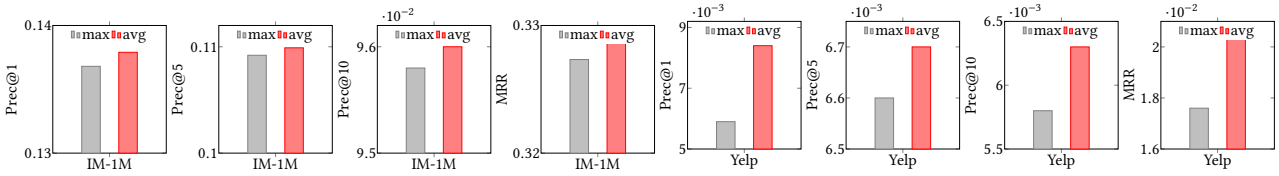

Figure 4: The impacts of different pooling strategies for RKGE on IM-1M and Yelp datasets.

by the proposed RKGE model. To understand the effect of different paths in characterizing user preferences, we compare two pooling strategies: 1) max-pooling, which focuses on the most important paths and 2) average-pooling, which aims at aggregating impacts of all paths. Hence, average-pooling avoids the result being dominated by a certain path. Figure 4 illustrates their performance on the two datasets, from which we can note that the performance of average-pooling is generally better than that of max-pooling. This supports our intuition that user preferences towards items are determined by combinations of heterogeneous factors and highlights the importance of a method that can make full use of these factors.

Interpretability of RKGE. By fully capturing the semantics of entities and entity relations encoded in KGs, RKGE can not only provide better recommendation, but also possess a better interpretability for user-item interactions. To demonstrate this, we first map for each user her rated items (i.e., items in training data) and the correctly recommended items (i.e., the intersection between items in her top-10 recommendation list and test data) into the KG, and then check whether there are semantic paths linking those items. For conciseness, here we only show the results of a randomly sampled user, referred to as Bob, in Figure 5. Similar observations can be obtained for other users on the two datasets.

In Figure 5, the items on the top are Bob's rated items in the training data, whereas the items on the bottom are correctly recommended ones in the test data. For simplicity, we only keep four types of entities in the KG, i.e., movie, genre, actor and director. Several interesting findings are obtained. First, the correctly recommended movies are all connected to Bob's rated movies either by genre (e.g., "Low Down Dirty Shame" – "Star Wars"), actors (e.g., "Air Force One" – "The Devil's Own"), or directors (e.g., "Raiders of Lost Ark" – "Jaws"). This suggests that RKGE can well infer Bob's specific preference from different angles with KGs, based on which generates correct recommendations. Second, most of the correctly recommended movies are connected with rated movies by different types of paths, instead of a single one. For example, "Raiders of Lost Ark" connects "Jaws" by "Adventure" and "Steven Spielberg"; "Air Force One" links with "Star Wars" by "Action", "Adventure" and "Harrison Ford". This implies that user-item interactions are

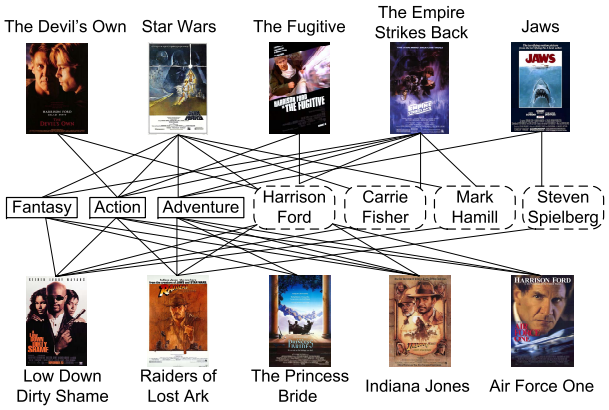

Figure 5: A running example to help illustrate the interpretability of RKGE on IM-1M dataset.

co-influenced by different paths, possibly with different degrees, and that their joint effects can be effectively captured by RKGE, allowing it to provide recommendations with high interpretability.

4.3 Comparative Results

Table 3 summarizes the performance of all comparison methods across the two real-world datasets, where two views are created for each dataset: 'All Users' indicates all the users are considered in the test data; while 'Cold Start' implies only users with less than 5 ratings are involved in the test data. A number of interesting observations can be noted from the results for the two views.

Performance on All Users. As the basic recommendation approaches considering no auxiliary information, MostPop and BPRMF perform worse than other methods. This helps confirm the usefulness of KGs for recommendation. We further observe that BPRMF highly outperforms MostPop, which is mainly because BPRMF is a personalized recommendation method via learning individual user's preference, whereas MostPop is a simple and non-personalized one.

By incorporating item attributes in the KG as raw features, LIBFM performs better than BPRMF, sometimes even better than existing KG based methods (e.g., HeteRS, HeteRec on Yelp), suggesting its

Table 3: Performance of all comparison approaches on IM-1M and Yelp across all the evaluation metrics. The best performance is boldfaced; the runner up is labeled with '*'; the column 'Improve' indicates the relative improvements that RKGE achieves w.r.t. the best performance of methods proposed by others.

View	Datasets	Metrics	MostPop	BPRMF	LIBFM	NCF	HeteRS	HeteRec	GraphLF	CKE	RKGE	Improve
All Users	IM-1M	Prec@1	0.0118	0.0409	0.0459	0.0450	0.0689	0.0764	0.1069*	0.0954	**0.1396**	30.58%
		Prec@5	0.0064	0.0438	0.0525	0.0482	0.0528	0.0579	0.0360	0.0781*	**0.1092**	39.82%
		Prec@10	0.0081	0.0441	0.0456	0.0485	0.0475	0.0488	0.0581	0.0682*	**0.0861**	26.25%
		MRR	0.0245	0.1234	0.1412	0.1360	0.1600	0.1737	0.1524	0.2440*	**0.3056**	25.25%
	Yelp	Prec@1	0.0003	0.0051	0.0054	0.0056	0.0047	0.0072	0.0083	0.0084*	**0.0113**	34.52%
		Prec@5	0.0007	0.0058	0.0059	0.0060*	0.0052	0.0050	0.0054	0.0057	**0.0070**	16.67%
		Prec@10	0.0004	0.0052	0.0054	0.0055	0.0031	0.0039	0.0056*	0.0053	**0.0062**	10.71%
		MRR	0.0010	0.0162	0.0167	0.0178	0.0116	0.0151	0.0189*	0.0178	**0.0218**	15.34%
Cold Start	IM-1M	Prec@1	0.0028	0.0171	0.0330	0.0188	0.0405	0.0573	0.0677	0.0687*	**0.0809**	17.76%
		Prec@5	0.0017	0.0191	0.0203	0.0210	0.0428	0.0428	0.0267	0.0432*	**0.0481**	11.34%
		Prec@10	0.0013	0.0205	0.0273	0.0225	0.0370	0.0402	0.0422*	0.0372	**0.0467**	10.66%
		MRR	0.0050	0.0438	0.0457	0.0481	0.1239	0.1355	0.1188	0.1408*	**0.1521**	8.03%
	Yelp	Prec@1	0.0004	0.0028	0.0042	0.0031	0.0037	0.0053	0.0061	**0.0067**	0.0061*	-8.96%
		Prec@5	0.0003	0.0037	0.0040	0.0041	0.0035	0.0035	0.0038	0.0052*	**0.0056**	7.69%
		Prec@10	0.0003	0.0031	0.0039	0.0034	0.0031	0.0032	0.0047*	0.0043	**0.0055**	17.02%
		MRR	0.0006	0.0098	0.0104	0.0108	0.0097	0.0113	0.0141	**0.0151**	0.0149*	-1.32%

superiority in utilizing auxiliary information for effective recommendation. Despite this, LIBFM models entity interactions in a linear fashion, thus is intrinsically limited by its expressive power for capturing complex patterns. Well-designed neural network based methods are more capable of modeling complex entity relations, as shown by the performance of NCF. Although merely considering user-item interaction data, NCF sometimes even performs better than LIBFM, demonstrating the effectiveness of neural architecture.

In terms of the methods specifically designed for KGs, HeteRec outperforms HeteRS. The possible reason is that HeteRS is a graph based method built upon random walk, thus failing to explicitly capture the semantics of entities and entity relations encoded in KG, whereas HeteRec is a latent factor model based approach which exploits the relation heterogeneity in the KG by introducing meta paths. This verifies that semantic paths in KG indeed facilitate to generate effective recommendation. GraphLF is also based on random walk, yet it combines the strength of latent factorization and logic reasoning, thus achieving better performance than HeteRS and HeteRec. By learning item semantic representations from KGs, CKE generally performs the best among the four existing KG based methods (i.e., HeteRS, HeteRec, GraphLF and CKE), implying the effectiveness of network embedding for better recommendation. However, it ignores the relations of paired entities linked by paths, thus failing to capture the full semantics encoded by KGs.

Overall, when compared with all the other comparison methods across the two datasets, our proposed RKGE consistently achieves the best performance. The improvements w.r.t. Precision and MRR are 26.42%, 20.30% on average, respectively (Paired t-test, p-value < 0.01). This implies that the recommendation performance can be further boosted by appropriately combining the strengths of network embedding and semantic path mining on KGs.

Performance on Cold Start. Similar observations with "All Users" can be seen on "Cold Start". As in the previous setting, RKGE significantly outperforms (p-value < 0.01) the best existing method by 9.25% and 3.36% for Precision and MRR on IM-1M, respectively. While on Yelp, the case is slightly different. The performance of RKGE is worse than CKE on some metrics (e.g., Prec@1, MRR). This is possibly due to the extremely low graph density of Yelp compared with IM-1M, leading to insufficient semantic paths for RKGE to take advantage of.

We further analyze the robustness of the compared methods for cold start recommendation. We observe that most methods are vulnerable to cold start users. This is because these methods learn user preference based on historical interactions, which contains very limited information for cold start users. Interestingly, CKE and RKGE consistently outperform other methods, implying the robustness of KG embedding for cold start users. RKGE further outperforms CKE, showing that the usage of path semantics by RKGE can effectively capture users' preferences even from their limited historical interaction records. Besides, we also notice that the improvement ratios of RKGE on "All Users" are larger than those on "Cold Start". This could be explained by the fact that different from other cold-start cases where cold-start users possess similar amount of external information as warm-start users, in recommendation with KGs, cold-start users have very limited number of paths linking entities in KGs. Therefore, the recommendation with KGs for cold-start users is more difficult.

5 CONCLUSION

Knowledge graphs have attracted a considerable amount of interest from the recommendation community due to its effectiveness in boosting recommendation performance. This paper presents a knowledge graph embedding framework – RKGE – with a novel recurrent network architecture for high-quality recommendation. RKGE not only learns the semantic representation of different types of entities but also automatically captures entity relations encoded in KGs. Extensive validation on two real-world datasets demonstrates the superiority of RKGE against the state-of-the-art recommendation methods. For future work, we plan to extend RKGE by considering the taxonomy of entity types in KGs.

REFERENCES

[1] Phil Blunsom, Edward Grefenstette, and Nal Kalchbrenner. 2014. A Convolutional Neural Network for Modelling Sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics.

[2] Rose Catherine and William Cohen. 2016. Personalized Recommendations using Knowledge Graphs: A Probabilistic Logic Programming Approach. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys)*. ACM, 325–332.

[3] Sneha Chaudhari, Amos Azaria, and Tom Mitchell. 2017. An Entity Graph based Recommender System. *AI Communications* 30, 2 (2017), 141–149.

[4] Tianqi Chen, Weinan Zhang, Qiuxia Lu, Kailong Chen, Zhao Zheng, and Yong Yu. 2012. SVDFeature: A Toolkit for Feature-based Collaborative Filtering. *Journal of Machine Learning Research (JMLR)* 13, Dec (2012), 3619–3622.

[5] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (almost) from Scratch. *Journal of Machine Learning Research (JMLR)* 12, Aug (2011), 2493–2537.

[6] Francois Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. 2007. Random-walk Computation of Similarities Between Nodes of a Graph with Application to Collaborative Recommendation. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 19, 3 (2007), 355–369.

[7] László Grad-Gyenge, Peter Filzmoser, and Hannes Werthner. 2015. Recommendations on a Knowledge Graph. In *1st International Workshop on Machine Learning Methods for Recommender Systems, MLRec*. 13–20.

[8] Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 855–864.

[9] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web (WWW)*. International World Wide Web Conferences Steering Committee, 173–182.

[10] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. 2010. Multiverse Recommendation: N-dimensional Tensor Factorization for Context-aware Collaborative Filtering. In *Proceedings of the 4th ACM Conference on Recommender Systems (RecSys)*. ACM, 79–86.

[11] Yann LeCun, Yoshua Bengio, et al. 1995. Convolutional Networks for Images, Speech, and Time Series. *The handbook of Brain Theory and Neural Networks* 3361, 10 (1995), 1995.

[12] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *Proceedings of 29th AAAI Conference on Artificial Intelligence (AAAI)*, Vol. 15. 2181–2187.

[13] Zemin Liu, Vincent W Zheng, Zhou Zhao, Fanwei Zhu, Kevin Chen-Chuan Chang, Minghui Wu, and Jing Ying. 2017. Semantic Proximity Search on Heterogeneous Graph by Proximity Embedding. In *Proceedings of 31st AAAI Conference on Artificial Intelligence (AAAI)*.

[14] Zemin Liu, Vincent W Zheng, Zhou Zhao, Fanwei Zhu, Kevin Chen-Chuan Chang, Minghui Wu, and Jing Ying. 2018. Distance-aware dag embedding for proximity search on heterogeneous graphs. In *Proceedings of 32nd AAAI Conference on Artificial Intelligence (AAAI)*. AAAI.

[15] Chen Luo, Wei Pang, Zhe Wang, and Chenghua Lin. 2014. Hete-cf: Social-based Collaborative Filtering Recommendation using Heterogeneous Relations. In *Data Mining (ICDM), 2014 IEEE International Conference on*. IEEE, 917–922.

[16] Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic Matrix Factorization. In *Advances in Neural Information Processing Systems*. 1257–1264.

[17] Wenjie Pei, Tadas Baltrušaitis, David MJ Tax, and Louis-Philippe Morency. 2017. Temporal Attention-gated Model for Robust Sequence Classification. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE, 820–829.

[18] Wenjie Pei, Jie Yang, Zhu Sun, Jie Zhang, Alessandro Bozzon, and David MJ Tax. 2017. Interacting Attention-gated Recurrent Networks for Recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM)*. ACM, 1459–1468.

[19] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online Learning of Social Representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 701–710.

[20] Tuan-Anh Nguyen Pham, Xutao Li, Gao Cong, and Zhenjie Zhang. 2015. A General Graph-based Model for Recommendation in Event-based Social Networks. In *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*. IEEE, 567–578.

[21] Tuan-Anh Nguyen Pham, Xutao Li, Gao Cong, and Zhenjie Zhang. 2016. A General Recommendation Model for Heterogeneous Networks. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 28, 12 (2016), 3140–3153.

[22] M Ross Quillan. 1968. Semantic Memory. In *Semantic Information Processing*.

[23] Steffen Rendle. 2010. Factorization Machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. IEEE, 995–1000.

[24] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI)*. AUAI Press, 452–461.

[25] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International Conference on World Wide Web (WWW)*. ACM, 285–295.

[26] Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional Recurrent Neural Networks. *IEEE Transactions on Signal Processing* 45, 11 (1997), 2673–2681.

[27] Chuan Shi, Jian Liu, Fuzhen Zhuang, S Yu Philip, and Bin Wu. 2016. Integrating Heterogeneous Information via Flexible Regularization Framework for Recommendation. *Knowledge and Information Systems* 49, 3 (2016), 835–859.

[28] Chuan Shi, Zhiqiang Zhang, Ping Luo, Philip S Yu, Yading Yue, and Bin Wu. 2015. Semantic Path based Personalized Recommendation on Weighted Heterogeneous Information Networks. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM)*. ACM, 453–462.

[29] Yue Shi, Martha Larson, and Alan Hanjalic. 2014. Collaborative Filtering Beyond the User-item Matrix: A Survey of the State of The Art and Future Challenges. *ACM Computing Surveys (CSUR)* 47, 1 (2014), 3.

[30] Ajit P Singh and Geoffrey J Gordon. 2008. Relational Learning via Collective Matrix Factorization. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 650–658.

[31] Alexander J Smola and Risi Kondor. 2003. Kernels and Regularization on Graphs. In *Learning Theory and Kernel Machines (COLT)*. Springer, 144–158.

[32] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research (JMLR)* 15, 1 (2014), 1929–1958.

[33] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S Yu, and Tianyi Wu. 2011. Pathsim: Meta Path-based Top-k Similarity Search in Heterogeneous Information Networks. *Proceedings of the VLDB Endowment* 4, 11 (2011), 992–1003.

[34] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*. 3104–3112.

[35] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale Information Network Embedding. In *Proceedings of the 24th International Conference on World Wide Web (WWW)*. International World Wide Web Conferences Steering Committee, 1067–1077.

[36] Ellen M Voorhees et al. 1999. The TREC-8 Question Answering Track Report. In *Trec*, Vol. 99. 77–82.

[37] Yueyang Wang, Yuanfang Xia, Siliang Tang, Fei Wu, and Yueting Zhuang. 2017. Flickr Group Recommendation with Auxiliary Information in Heterogeneous Information Networks. *Multimedia Systems* 23, 6 (2017), 703–712.

[38] Paul J Werbos. 1988. Generalization of Backpropagation with Application to a Recurrent Gas Market Model. *Neural networks* 1, 4 (1988), 339–356.

[39] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent Recommender Networks. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining (WSDM)*. ACM, 495–503.

[40] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *International Conference on Machine Learning (ICML)*. 2048–2057.

[41] Xiao Yu, Xiang Ren, Quanquan Gu, Yizhou Sun, and Jiawei Han. 2013. Collaborative Filtering with Entity Similarity Regularization in Heterogeneous Information Networks. *Proceedings of 5th IJCAI Workshop on Heterogeneous Information Network Analysis (IJCAI-HINA)* 27 (2013).

[42] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. 2014. Personalized Entity Recommendation: A Heterogeneous Information Network Approach. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining (WSDM)*. ACM, 283–292.

[43] Xiao Yu, Xiang Ren, Yizhou Sun, Bradley Sturt, Urvashi Khandelwal, Quanquan Gu, Brandon Norick, and Jiawei Han. 2013. Recommendation in Heterogeneous Information Networks with Implicit User Feedback. In *Proceedings of the 7th ACM conference on Recommender Systems (RecSys)*. ACM, 347–350.

[44] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative Knowledge base Embedding for Recommender Systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 353–362.

[45] Jing Zheng, Jian Liu, Chuan Shi, Fuzhen Zhuang, Jingzhi Li, and Bin Wu. 2017. Recommendation in Heterogeneous Information Network via Dual Similarity Regularization. *International Journal of Data Science and Analytics* 3, 1 (2017), 35–48.