Attentive Knowledge Graph Embedding for Personalized Recommendation

Xiao Sha
Nanyang Technological University
Singapore
SHAX0001@e.ntu.edu.sg

Zhu Sun*
Nanyang Technological University
Singapore
sunzhuntu@gmail.com

Jie Zhang
Nanyang Technological University
Singapore
ZhangJ@ntu.edu.sg

ABSTRACT

Knowledge graphs (KGs) have proven to be effective for high-quality recommendation. Most efforts, however, explore KGs by either extracting separate paths connecting user-item pairs, or iteratively propagating user preference over the entire KGs, thus failing to efficiently exploit KGs for enhanced recommendation. In this paper, we design a novel attentive knowledge graph embedding (AKGE) framework for recommendation, which sufficiently exploits both semantics and topology of KGs in an interaction-specific manner. Specifically, AKGE first automatically extracts high-order subgraphs that link user-item pairs with rich semantics, and then encodes the subgraphs by the proposed attentive graph neural network to learn accurate user preference. Extensive experiments on three real-world datasets demonstrate that AKGE consistently outperforms state-of-the-art methods. It additionally provides potential explanations for the recommendation results.

1 INTRODUCTION

Recently, knowledge graphs (KGs) [38] have attracted increasing attention in the community of recommender systems due to their comprehensive auxiliary data for effective recommendation, such as item attributes and user profiles [45, 52]. However, it is challenging to effectively integrate such heterogeneous information into recommender systems. Till now, two types of KG-aware recommendation algorithms have been broadly studied, namely path-based [52, 57] and propagation-based methods [45, 50]. The former methods model the user-item connectivity by extracting linear paths from KGs, and therefore fail to exploit the rich semantics and topology of KGs. The latter methods iteratively propagate user preference over the entire KGs, which inevitably introduces noise irrelevant to the specific user-item interaction and thus limits the performance.

Toy Example. To illustrate, Figure 1 depicts a toy example of a KG in business domain (i.e., Yelp). It shows fairly strong relations between McDonald's and KFC with various semantics: belong to the same category (i.e. Food); located in the same city (i.e. SunCity); and simultaneously rated by two friends (i.e. May and Amy). To infer Mike's preference to KFC, path-based methods first extract several qualified paths linking Mike and KFC from the KG with length constraint (e.g., no more than 3). Hence, four paths are extracted: (1) Mike → McDonald's → May → KFC; (2) Mike → McDonald's → Amy → KFC; (3) Mike → McDonald's → SunCity → KFC; and (4) Mike → McDonald's → Food → KFC. Then each extracted path is modeled independently, and their impacts are only aggregated at the final stage for estimating Mike's taste over KFC. Nevertheless, they overlook that the strong relations between Mike and KFC,

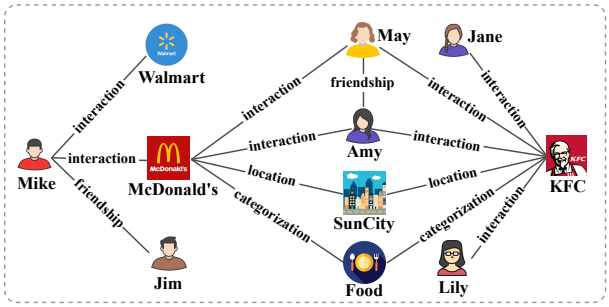


Figure 1: A toy example of KG in Yelp, which contains users, business, categories, cities as entities; interaction, friendship, location, categorization as the entity relations.

originally expressed by the graph, may not be well reflected by the simple aggregation of the separate paths. That is to say, such weak coupling hinders the paths from getting a more complete picture of the complicated topology between Mike and KFC [29]. In addition, the longer path Mike → McDonald's → May → Amy → KFC, indicating the friendship of May and Amy, is simply ignored due to the length constraint. Such information loss further weakens the strong connections between Mike and KFC, thus leading to inaccurate recommendations. In contrast, propagation-based methods model the interaction between Mike and KFC by iteratively propagating information over the entire KG. The preference of Mike is learned by aggregating information from all of its neighbors (e.g., Walmart, McDonald's and Jim). Nevertheless, such information may contain noise (e.g., Walmart and Jim) that is irrelevant to this specific interaction, which misleads the inference of Mike's interest on KFC.

To ease the limitations of path-based and propagation-based methods, we believe it is of critical importance to sufficiently exploit KGs in an interaction-specific manner. To this end, we exploit subgraphs that connect user-item pairs to characterize their connectivities, which benefits recommendation in two aspects: (1) as a nonlinear combination of separate paths, the subgraph contains both the rich semantics and topology of KGs, and thus is more expressive than linear paths; and (2) the subgraph only preserves entities and relations that are relevant to the specific user-item interaction; it can thus efficiently avoid noise from the entire KG. It is, however, non-trival to employ subgraphs for user preference inference with the following two major challenges.

The first challenge is how to mine semantic subgraphs for more comprehensive descriptions on entity relations (e.g., the relations between Mike and KFC). Due to the large volume of KGs, it is not practical to apply traditional methods with expensive computational cost on KGs, such as breadth first search (BFS) [60] and

*Zhu Sun is the corresponding author.

depth first search (DFS) [41][1]. We, therefore, propose to convert the time-consuming subgraph mining into labor-saving path sampling, and then construct the subgraph by assembling the sampled paths. A novel *distance-aware sampling strategy* is thus devised to help select salient paths between user-item pairs, so as to blackuce the noise that may be introduced in the subgraph.

The second challenge is how to encode the extracted heterogeneous subgraphs in an effective fashion for enhanced user preference inference. The unique non-Euclidean structure of subgraphs motivates us to employ the powerful graph representation technique – graph neural networks (GNNs) [13, 24, 27, 35], which learn accurate embeddings for entities by iteratively aggregating information from neighbors in the high-order graph topology. However, all neighbors are treated equally in vanilla GNNs. Due to the heterogeneity of KGs, an entity may connect with multiple types of neighbors via various relations. To differentiate the impact of different neighbors, we design an attentive graph neural network with *relation-aware propagation* and *attentive aggregation* to help attentively aggregate more reliable information from neighbors for a better embedding learning.

In sum, we propose a novel end-to-end neural recommendation framework, named *Attentive Knowledge Graph Embedding* framework – AKGE. Specifically, the distance-aware sampling strategy helps mine high-order semantic subgraphs, representing complex relations between user-item pairs, in an automatic manner; and the attentive graph neural network aims to model the complicated user-item connectivity (i.e., the high-order subgraphs between user-item pairs) by considering the heterogeneity of KGs. To the best of our knowledge, this is the first time that heterogeneous subgraphs have been used to characterize user-item interactions for KG-aware recommendation. Extensive experiments on three real-world datasets demonstrate that AKGE significantly beats state-of-the-art methods with a lift of 9.30%, 9.54% w.r.t. Hit and NDCG on average, respectively. In addition, the elaborated case study helps illustrate the interpretability of AKGE for effective recommendation.

2 RELATED WORK

This section gives a review of existing KG-based recommendation methods, which can be generally classified into three categories:

Direct-relation based Methods. A line of research leverages the relations of directly connected entities in KGs for embedding learning. For instance, CKE [58] adopts TransR [28] to learn item embeddings with the involvement of KGs. DKN [46] generates news embeddings by utilizing KGs via TransD [22]. KSR [20] exploits KGs to boost sequential recommendations via TransE [2]. More recently, KTUP [3] jointly learns the recommendation model and KG completion via TransH [53]. RCF [55] jointly models user preference via attention mechanisms and item relations via DistMult [56]. MKR [48] utilizes KG embeddings to regularize recommendation tasks. Though significant improvements have been achieved, these methods cannot fully capture the complex semantics of user-item connectivity for enhanced embedding learning, as they only consider the direct relations between entities.

[1]Both BFS and DFS take an $O(|\mathcal{E}| + |\mathcal{L}|)$ complexity, where $|\mathcal{E}|$ and $|\mathcal{L}|$ denote the sizes of entity set and link set of KGs, respectively.

Semantic-path based Methods. Many methods measure user-item connectivity via meta-path based similarity [39], such as PER [57], HeteCF [31] and SemRec [37]. Others leverage the meta-path based random walk for better embedding learning, such as HERec [36], HIN2Vec [12], HINE [21] and Metapath2vec [8]. The dependency on handcrafted features dramatically hinders the generality of these methods. Hence, RKGE [40] and KPRN [52] automatically extract paths (with length constraint) connecting user-item pairs, and then model these paths via RNNs. However, decomposing the sophisticated user-item connectivity into separate linear paths would inevitably lead to information loss. In contrast, our AKGE exploits rich semantics of user-item connectivity via expressive subgraphs, in an automatic manner.

Propagation based Methods. Recent methods iteratively perform propagation over the whole KG to assist in recommendation. For instance, RippleNet [44, 45] extends the user's interests along KG links to discover her potential interests. KGCN [49] and KGCN-LS [47] apply graph convolutional network (GCN) [13] to compute embeddings of items via propagation among their neighbors in KGs. Most recently, KGAT [50] recursively performs propagation over KGs via GCN to refine entity embeddings. Taking the whole KG as input, these methods learn embedding for each entity by aggregating information from all of its neighbors. Noise (i.e., the less informative neighbors) and heavy computation (i.e., the large number of neighbors) would be inevitably introduced into the embedding learning process, as there is no constraint between the target entity and its neighbors, thus hurting the performance. In contrast, AKGE preserves only the highly related entities and relations for a specific user-item pair by mining the semantic subgraph connecting them from KGs, and thus is able to learn more accurate embeddings for user preference inference.

3 ATTENTIVE KNOWLEDGE GRAPH EMBEDDING

This section presents the proposed framework – attentive knowledge graph embedding (AKGE), which exploits the expressive subgraphs for KG-based recommendation. The overall framework of AKGE is illustrated by Figure 2, composed of three modules: 1) Subgraph Construction – it automatically mines semantic subgraphs to represent the high-order user-item connectivity based on a distance-aware path sampling strategy; 2) Attentive Graph Neural Network (AGNN) – the subgraphs are further encoded by a novel attentive graph neural network to learn semantic embeddings for entities; 3) Multi-layer Perceptron (MLP) Prediction – by feeding the well-learned user and item embeddings from AGNN, it uses nonlinear layers to predict user preferences towards items.

Notations. We denote the user and item sets as $\mathcal{U} = \{u_1, u_2, \cdots, u_n\}$ and $\mathcal{I} = \{i_1, i_2, \cdots, i_m\}$, respectively. Each entry $r_{u,i}$ in the user-item feedback matrix $\mathbf{R} \in \mathbb{R}^{n \times m}$ is defined as: $r_{u,i} = 1$ if the interaction between user u and item i is observed, and 0 otherwise. For generality, we use "entity" to refer to objects (e.g., user, business, category, and city) that can be mapped into a KG (denoted as $\mathcal{G}$). The definitions of KGs and the investigated task are given below.

Definition 1. Knowledge Graph. Let $\mathcal{E}, \mathcal{L}$ denote the sets of entities and links, respectively. A KG is defined as an undirected graph $\mathcal{G} = (\mathcal{E}, \mathcal{L})$ with entity type and link type mapping functions

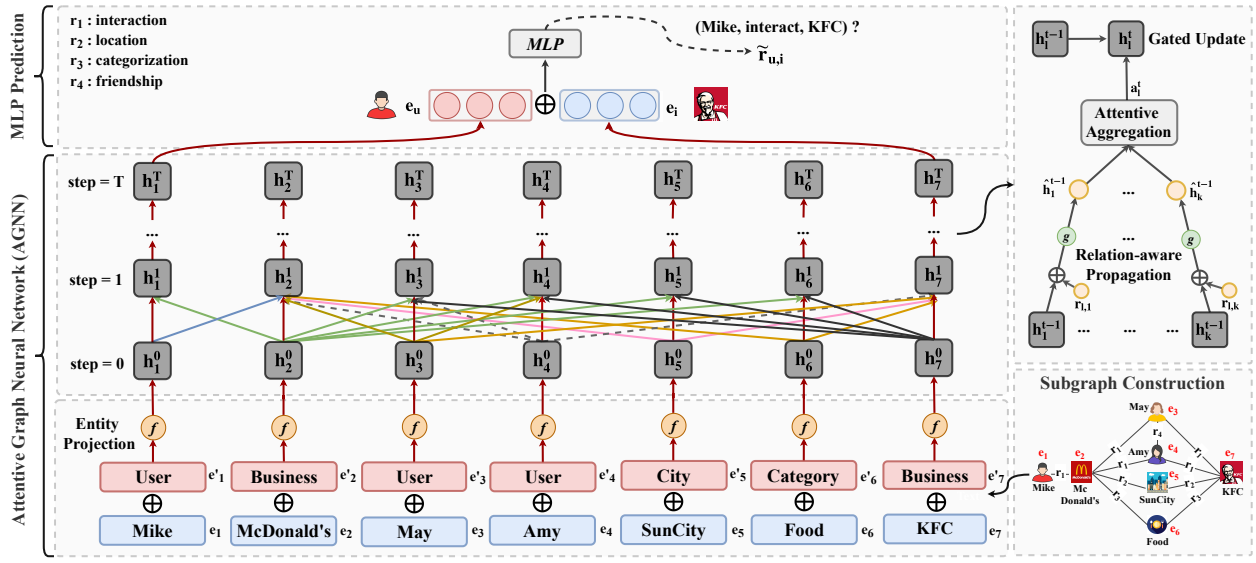


Figure 2: The overall framework of AKGE describing the case of a user-item pair, which consists of three modules: Subgraph Construction, AGNN and MLP Prediction. The subfigure in the upper right presents three of the components of AGNN: Relation-aware Propagation, Attentive Aggregation and Gated Update.

$\phi : \mathcal{E} \rightarrow \mathcal{A}$ and $\varphi : \mathcal{L} \rightarrow \mathcal{R}$. 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 $\varphi(l) \in \mathcal{R}$. The types of entities $|\mathcal{A}| > 1$ or the types of links $|\mathcal{R}| > 1$ in KGs.

Definition 2. KG-based Top-N Recommendation. Given the KG $\mathcal{G}$, for each user $u \in \mathcal{U}$, our task is to generate a ranked list of items that will be of interest to user u.

3.1 Subgraph Construction

It is computationally prohibitive to construct subgraphs between user-item pairs via traditional graph mining methods, such as BFS [60] and DFS [41], over the whole KG. To reduce the complexity, we thus propose a more efficient subgraph construction strategy, which converts the time-consuming subgraph mining into salient path sampling and then reconstructs the subgraph by assembling all sampled paths between the user-item pairs.

Distance-aware Path Sampling. Existing KG-based methods [6, 8, 12, 21, 36] mainly adopt the meta-path guided random walk to generate paths between user-item pairs, which requires prior knowledge from domain experts for optimal meta-path selection [6, 8, 36], and only keeps paths shorter than a predefined length [12, 21]. To ease these limitations and further reduce noise, we devise a *distance-aware path sampling strategy* that automatically selects salient paths. We presume that a local neighbor with shorter distance can help reflect more reliable connections. Hence, in our strategy, the next step of a walker is determined by the distance between the current entity and candidate entities.

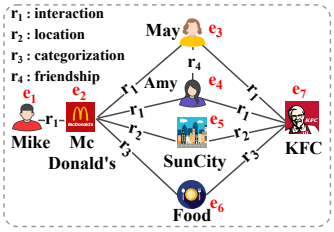
Inspired by translational distance models [2, 28, 53], we project all the entities into Euclidean space as embeddings, and measure their distances via Euclidean distance of the corresponding embeddings, which obeys the triangle inequality [18]. Meanwhile, enlightened by the tricks for training neural networks in [15, 20], we pre-train the embeddings of entities in the KG by TransR [28].[2] By doing this, we can calculate the pairwise distance for any two consecutive entities along a path and sum all these distances as the distance of the entire path. Finally, only the K paths with the shortest distance are preserved for constructing semantic subgraphs.

Path Assembling. Guided by the sampling strategy, a set of paths between a user-item pair – $\mathcal{P}(u, i)$ are sampled. By assembling paths in $\mathcal{P}(u, i)$, we generate the semantic subgraph, represented by an adjacency matrix – $\mathbf{A}$. To be specific, we traverse $\mathcal{P}(u, i)$ to map each object and connection along a path into the subgraph as entities and links, respectively. Take the path: Mike $\rightarrow$ McDonald's $\rightarrow$ May $\rightarrow$ KFC in Figure 3(a) as an example. Objects Mike, McDonald's, May, KFC are mapped as entities: e_1, e_2, e_3, e_7; connections between entities (e.g., Mike and McDonald's) are mapped as relations (e.g., r_1) being represented as entries (e.g., $a_{1,2} = a_{2,1} = 1$) in $\mathbf{A}$, as shown in Figure 3(b). The adjacency matrix $\mathbf{A}$ that encodes the topology of the subgraph is then fed into the AGNN module to guide information propagation among entities. To note with, the subgraph is treated as an undirected graph as the connected entities would exert influences on each other along the link.

3.2 Attentive Graph Neural Network

To efficiently exploit KGs for recommendation, we take inspiration from the recent advance – gated graph neural network (GGNN) [27], which has attracted considerable attention due to its superior learning capability on graphs, in various domains [25, 42, 51]. Typically, GGNN mainly focuses on dealing with homogeneous graphs. We thus propose an attentive graph neural network (AGNN), that tailors GGNN via fully considering the heterogeneity of KGs. It is achieved by four components, namely entity projection, relation-aware propagation, attentive aggregation and gated update.

[2] We empirically find out TransR outperforms other pre-train techniques in Section 4.4.



(a) An example of the subgraph

	e_1	e_2	e_3	e_4	e_5	e_6	e_7
e_1	0	1	0	0	0	0	0
e_2	1	0	1	1	1	1	0
e_3	0	1	0	1	0	0	1
e_4	0	1	1	0	0	0	1
e_5	0	1	0	0	0	0	1
e_6	0	1	0	0	0	0	1
e_7	0	0	1	1	1	1	0

(b) Adjacency matrix (A)

Figure 3: Toy example for path assembling.

Entity Projection. Given a subgraph $\mathcal{G}_s = \{\mathcal{E}_s, \mathcal{L}_s\}$ with its adjacency matrix $\mathbf{A}_s$ as input, we first adopt an embedding look-up layer to project each entity e_l (e.g., KFC) and its corresponding entity type e'_l (e.g., business) into low dimensional vectors $\mathbf{e}_l \in \mathbb{R}^d$ and $\mathbf{e}'_l \in \mathbb{R}^{d'}$, where d and d' are the respective embedding sizes. Distinct from vanilla GNN only taking entity embeddings as inputs, we additionally integrate the embeddings of entity types to better encode the heterogeneity of KGs [52]. In particular, for each entity $e_l \in \mathcal{E}_s$, we enhance its original entity embedding with its type embedding, given by:

$$\hat{\mathbf{e}}_l = f(\mathbf{e}_l \oplus \mathbf{e}'_l) \tag{1}$$

where $\hat{\mathbf{e}}_l$ is the type-enhanced embedding of e_l; $f(\mathbf{x}) = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$; $\mathbf{W}, \mathbf{b}$ are the transformation matrix and bias term, respectively; $\oplus$ means concatenation operation. The type-enhanced embedding of e_l is used to initialize its hidden state at propagation step $t = 0$,

$$\mathbf{h}^0_l = \hat{\mathbf{e}}_l \tag{2}$$

In this way, AGNN incorporates the heterogeneous type information into the learned entity embeddings, encoding more information from KGs. Moreover, as parameter initialization is vital for the performance of deep learning models [10], we initialize $\mathbf{e}_l$ with the pre-trained embeddings via TransR [28].

Relation-aware Propagation. Given $\mathbf{h}^0_l$, AGNN updates embedding for e_l by propagating and aggregating information from its neighbors. The vanilla GGNN merely propagates information w.r.t. entity embeddings, while ignores the semantic information encoded in entity relations. In KGs, an entity may connect with multiple neighbors via various relations (e.g. friendship, interaction, location), indicating different semantics. The rich semantics encoded in relations are critical for understanding the diverse user intents [52]. Hence, we propose a *relation-aware propagation* to explicitly fuse the heterogeneous relations of KGs for better embedding learning. In particular, we represent the relation between the target entity e_l and its neighbor e_k as $r_{l,k}$, which is utilized to enhance the hidden state of its neighbor e_k, achieved by:

$$\hat{\mathbf{h}}^t_k = g(\mathbf{h}^t_k \oplus \mathbf{r}_{l,k}) \qquad e_k \in \mathcal{N}_l \tag{3}$$

where $\mathbf{h}^t_k$ is the hidden state of e_k at propagation step t; $\mathbf{r}_{l,k} \in \mathbb{R}^{d''}$ is the embedding of $r_{l,k}$; d'' is the embedding size; $g(\mathbf{x}) = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b})$;[3] $\mathcal{N}_l$ is the set of neighbors of e_l. The output $\hat{\mathbf{h}}^t_k$ is the relation-aware hidden state of e_k, which condenses the information propagated from the neighbor e_k to e_l along their relation $r_{l,k}$, and then will be aggregated to update the embedding of the target entity e_l. In sum, the relation-aware propagation enables AGNN to achieve a better exploitation on the heterogeneity of KGs.

Attentive Aggregation. Given the information propagated from neighbors, AGNN attentively aggregates the information for e_l via an attention mechanism. Formally, at each propagation step t, the target entity e_l aggregates the relation-aware hidden states of its neighbors, expressed as:

$$\mathbf{a}^t_l = (\mathbf{A}_{s_l} \odot \mathbf{Q}^t_l) \left[\hat{\mathbf{h}}^{t-1}_1, \cdots, \hat{\mathbf{h}}^{t-1}_{|\mathcal{E}_s|}\right]^\top + \mathbf{b} \tag{4}$$

where $\mathbf{a}^t_l$ is the attentive aggregation of neighbors' hidden states; $\mathbf{A}_{s_l} \in \mathbb{R}^{|\mathcal{E}_s|}$ is the row vector of $\mathbf{A}_s$ related to e_l, denoting the connections of e_l with all entities in $\mathcal{G}_s$; $|\mathcal{E}_s|$ is the total number of entities in $\mathcal{G}_s$; $\mathbf{Q}^t_l \in \mathbb{R}^{|\mathcal{E}_s|}$ is the attention weight vector, which discriminates the influence of local neighbors on e_l. We will introduce how to calculate the attention weight later; $\odot$ denotes Hadamard product; $(\mathbf{A}_{s_l} \odot \mathbf{Q}^t_l)$ thus denotes the weighted connections between e_l and its neighbors; for entities that have no connection with e_l, the corresponding entries in $\mathbf{A}_{s_l}$ and $\mathbf{Q}^t_l$ will be set 0; $\left[\hat{\mathbf{h}}^{t-1}_1, ..., \hat{\mathbf{h}}^{t-1}_{|\mathcal{E}_s|}\right]$ are relation-aware hidden states of all entities in $\mathcal{G}_s$ generated from the previous propagation step $(t-1)$; To simplify the operation of matrix multiplications, we generate the relation-aware hidden states of all the entities in $\mathcal{G}_s$ with the padded relation.

Now we present how to calculate the attention weight vector $\mathbf{Q}^t_l$ in Equation 4, which helps discriminate the influence of local neighbors on e_l. In vanilla GGNN, it assumes that all neighbors contribute equally to learn embedding for the target entity e_l, which however might introduce noise and mislead the embedding learning process [11, 30]. We, therefore, distinguish the subtle difference of its neighbors (e.g., e_k) via an attention mechanism, implemented with a two-layer neural network:

$$\alpha^t_{l,k} = \mathbf{w}^{\mathrm{T}}_2 \cdot (\mathbf{W}_1 \cdot [\hat{\mathbf{h}}^{t-1}_k \oplus \mathbf{h}^{t-1}_l] + \mathbf{b}_1) + b_2 \tag{5}$$

where $\alpha^t_{l,k}$ is the attention score of e_k at step t w.r.t. e_l; $\hat{\mathbf{h}}^{t-1}_k$ is the relation-aware hidden state of neighbor e_k; $\mathbf{h}^{t-1}_l$ is the previous hidden state of e_l. We then normalize the above attention scores across all neighbors of e_l by adopting the softmax function:

$$q^t_{l,k} = \frac{exp(\alpha^t_{l,k})}{\sum_{e_j \in \mathcal{N}_l} exp(\alpha^t_{l,j})} \tag{6}$$

where $q^t_{l,k}$ is the final attention weight, representing the strength of influence that e_k has on e_l. It makes up the weight vector $\mathbf{Q}^t_l$. By assigning the informative neighbor with a higher weight, the attention mechanism is capable of suggesting which neighbors to focus on, so as to capture the salient semantic relations. The attentive aggregation of neighbors' information $\mathbf{a}^t_l$ is then utilized as the input state to update embedding for e_l, as introduced next.

Gated Update. Given $\mathbf{a}^t_l$, AGNN updates the embedding for the target entity e_l through a gating mechanism similar to the gated recurrent units [7] for better controlling the information. Specifically,

[3] We also implement $g(\mathbf{x})$ with an MLP, but achieve unsatisfying performance.

two gates are obtained to regulate the flow of information,

$$\mathbf{z}_l^t = \sigma\left(\mathbf{W}_z \mathbf{a}_l^t + \mathbf{U}_z \mathbf{h}_l^{t-1}\right) \tag{7}$$

$$\mathbf{r}_l^t = \sigma\left(\mathbf{W}_r \mathbf{a}_l^t + \mathbf{U}_r \mathbf{h}_l^{t-1}\right) \tag{8}$$

where $\mathbf{z}_l^t$ and $\mathbf{r}_l^t$ are the update and reset gates, respectively; σ is the sigmoid function; $\mathbf{W_z}$, $\mathbf{U_z}$, $\mathbf{W_r}$, $\mathbf{U_r}$ are the learned weight matrices; $\mathbf{a}_l^t$ is the input state of e_l generated by Equation 4. Thus, the candidate hidden state of e_l ($\widetilde{\mathbf{h}}_l^t$) is constructed by the input state, previous hidden state and reset gate:

$$\widetilde{\mathbf{h}}_l^t = \tanh\left(\mathbf{W}_h \mathbf{a}_l^t + \mathbf{U}_h\left(\mathbf{r}_l^t \odot \mathbf{h}_l^{t-1}\right)\right) \tag{9}$$

where $\mathbf{W}_h, \mathbf{U}_h$ are the coefficient matrices; the reset gate $\mathbf{r}_l^t$ controls how much information from the previous hidden state $\mathbf{h}_l^{t-1}$ to be discarded. The current hidden state of e_l ($\mathbf{h}_l^t$) is then updated as the combination of the previous hidden state and the candidate hidden state, controlled by the update gate:

$$\mathbf{h}_l^t = \left(1 - \mathbf{z}_l^t\right) \odot \mathbf{h}_l^{t-1} + \mathbf{z}_l^t \odot \widetilde{\mathbf{h}}_l^t \tag{10}$$

By recursively aggregating information from neighbors, AGNN can achieve a better exploitation on not only the first-order connectivity of entities with a single propagation step ($t = 1$), but also the high-order connectivity with multiple propagation steps ($t > 1$). After iteratively propagating T steps, the final hidden states fuse the information of entities and their neighbors up to T hops away, which are utilized as entity embeddings for predicting user preferences. To sum up, AGNN is able to capture the high-order user-item connectivity via relation-aware propagation and attentive aggregation, endowing itself with expressive capability.

3.3 MLP Prediction

With AGNN, we can learn better embeddings for user u and item i, which are further utilized to predict user preference as below:

$$\widetilde{r}_{u,i} = MLP(\mathbf{e}_u \oplus \mathbf{e}_i) \tag{11}$$

where $\widetilde{r}_{u,i}$ is the estimated score for user u on item i; $\mathbf{e}_u, \mathbf{e}_i$ are the learned embeddings of u and i, respectively. In Equation (11), we feed the concatenated user and item embeddings into a MLP component to help model the complicated user-item interactions. Following the premise that neural networks can learn more abstractive features of data by using a small number of hidden units for higher layers [14], we empirically implement a tower structure for the MLP component, halving the layer size for each successive higher layer. We adopt ReLU [32] as the activation function for hidden layers and sigmoid function for the output layer to control the estimated score $\widetilde{r}_{u,i}$ into the range of $[0, 1]$.

3.4 Model Optimization

Objective Function. Following [15], we address the top-N recommendation task as a binary classification problem, where target value 1 means a user-item interaction is observed and 0 otherwise. Formally, we adopt the negative log-likelihood as the objective function, formulated by:

$$\mathcal{J} = -\sum_{(u,i)\in\mathcal{R}^+} \log \widetilde{r}_{u,i} + \sum_{(u,j)\in\mathcal{R}^-} \log(1 - \widetilde{r}_{u,j}) \tag{12}$$

Table 1: Statistics of the datasets.

		MI-1M	Last-FM	Yelp
User-Item Interactions	#Users	6,040	23,566	37,940
	#Items	3,382	48,123	11,516
	#Interactions	756,684	3,034,796	229,178
	#Data Density	3.704%	0.268%	0.052%
Knowledge Graph	#Entities	18,920	138,362	46,606
	#Relation Types	10	10	6
	#Links	968,038	2,007,423	302,937

Algorithm 1: AKGE Optimization

Input: $\mathcal{G}$, $\mathbf{R}$, K, T, L, d, d', d'', λ, γ, max_iter
1 Pre-train entity embeddings via TransR;
2 Initialize the embeddings of $e \in \mathcal{G}$ with pre-trained embeddings;
// Subgraph Construction
3 **foreach** *(u, i) pair in training set* **do**
4 Mine K paths $\mathcal{P}(u, i)$ by the distance-aware sampling strategy;
5 Assemble $\mathcal{P}(u, i)$ into subgraph $\mathcal{G}_s$;
6 **for** $iter = 1; iter \leq max\_iter; iter++$ **do**
7 **foreach** (u, i) *pair* **do**
// AGNN
8 **foreach** $e_l \in \mathcal{G}_s$ **do**
9 Initialize $\mathbf{h}_l^0$ based on Equation (1-2);
10 **for** $t = 1; t \leq T; t++$ **do**
11 Update $\mathbf{h}_l^t$ in parallel based on Equations (3-10);
// MLP Prediction
12 Calculate $\tilde{r}_{u,i}$ based on Equation (11);
13 Update parameters by back propagation through time;

where $\mathcal{R}^+$ and $\mathcal{R}^-$ denote the sets of observed and non-observed user-item interactions, respectively. We uniformly sample negative item j for each user u that she has not interacted with, and control the sampling ratio with regard to the number of positive item i to be 4:1. Parameters are learned by the back propagation through time (BPTT) [5] in AGNN module and by normal back-propagation in other modules. The optimization process is described by Algorithm 1, which is mainly composed of three parts: Subgraph Construction (lines 3-5), AGNN (lines 7-11) and MLP Prediction (line 12).

Complexity Analysis. The time cost of AKGE mainly comes from (a) subgraph construction and (b) subgraph encoding via AGNN. For (a), we can offline obtain the distance of all connected entities in the training data via the pre-trained embeddings. Given a target entity, we offline filter out all the out-going entities with the distance larger than a threshold, and build an alias table [26] for $O(1)$-time entity sampling. Hence, generating a subgraph for a user-item pair can be done in $O(PM + MlogM)$ offline time, where P is the path length and M is the maximum number of considered paths. In practice, $P \leq 5$ and $M \leq 1000$ [19, 52]. For (b), the computational complexity for the matrix multiplication is $\sum_{t=1}^{T} O(|\mathcal{E}_s|d^2)$, where d is the embedding size of hidden and input states, and T is the propagation step. The overall training complexity would be $O(|\mathbf{R}| \sum_{t=1}^{T} |\mathcal{E}_s|d^2)$, where $|\mathbf{R}|$ is the number of observed entries in $\mathbf{R}$. In practice, $T < |\mathcal{E}_s| \ll |\mathbf{R}|$. In sum, the complexity is linear to $|\mathbf{R}|$ and quadratic to the embedding size d.

4 EXPERIMENTS AND ANALYSIS

We conduct extensive experiments on three real-world datasetswith the goal of answering four research questions:

Table 2: Performance comparison on the three datasets. The best performance is boldfaced; the runner up is labeled with '*'; 'Improve' indicates the improvements (Paired t-test with $p-$value < 0.01) that AKGE achieves relative to the '*' results.

Data	Metrics	MostPop	UserKNN	ItemKNN	BPRMF	NeuMF	CKE	FMG	MCRec	RKGE	KPRN	KGAT	AKGE	Improve
MI-1M	Hit@10	0.4725	0.6375	0.6412	0.6837	0.7225	0.7264	0.5891	0.6988	0.7402	0.7831	0.8136*	**0.9035**	11.05%
	NDCG@10	0.2630	0.3701	0.3785	0.4316	0.4808	0.4893	0.3956	0.4479	0.5564	0.6285	0.6327*	**0.7096**	12.15%
Last-FM	Hit@10	0.4516	0.6382	0.6264	0.6652	0.7063	0.7104	0.6374	0.6092	0.7153	0.7462	0.8569*	**0.8961**	4.57%
	NDCG@10	0.2471	0.3625	0.3510	0.4042	0.4681	0.4715	0.3825	0.2764	0.5292	0.6014	0.6609*	**0.6987**	5.72%
Yelp	Hit@10	0.4236	0.5731	0.5936	0.6214	0.6719	0.6825	0.6107	0.5127	0.6857	0.7156	0.8059*	**0.8842**	9.72%
	NDCG@10	0.2128	0.3364	0.3450	0.3742	0.4527	0.4603	0.3584	0.2592	0.5081	0.5869	0.6317*	**0.6827**	8.07%

- **RQ1:** Does our proposed AKGE outperform the state of-the-art recommendation methods?
- **RQ2:** Can AKGE provide potential explanations about user preferences towards items?
- **RQ3:** How do our proposed path sampling strategy, relation-aware propagation and attentive aggregation affect the performance of AKGE, respectively?
- **RQ4:** How do different choices of hyper-parameters affect the preformance of AKGE?

4.1 Experimental Setup

Datasets. Three benchmark datasets are utilized: (1) **MovieLens-1M**[4] is a widely used dataset in movie recommendations [45], which describes user ratings towards movies ranging from 1-5; (2) **Last-FM**[5] is a music listening dataset collected from the Last.fm online music system, where the tracks are viewed as items. We use the same version of this dataset as in [50]; (3) **Yelp**[6] records user ratings on local business scaled from 1-5. Additionally, social relations as well as business attributes (e.g., category, city) are also included.

We process the datasets by following [4, 57]: if a user rates an item, we set it as an observed interaction with value 1 and 0 otherwise. Besides user-item interactions, we merge more information into KGs for each dataset. We combine MovieLens-1M with IMDb[7] as MI-1M by linking the titles and release dates of movies, so as to get side information about movies, such as genres, actors and directors. For Last-FM, we map tracks into objects in the database called Freebase via title matching to get attributes of tracks, such as artists, engineers, producers, versions, types, contained_songs, etc. For Yelp, we extract knowledge from the social network and local business information network (e.g., category, city). Table 1 summarizes the statistics of the three datasets.

Evaluation Protocols. We adopt *leave-one-out*, which has been widely used in the previous efforts [1, 15, 16, 33], to evaluate the recommendation performance. For each user, the latest interaction is held out as test set, and the remaining data is utilized as training set. Aligning with [9, 15, 52], during testing, for each user, we randomly sample 100 items that the user has not interacted with and then rank the test item among the 101 items, so as to reduce test complexity. Following [15, 52], we adopt Hit@N and NDCG@N as the evaluation metrics, compute both metrics for each test user and report the average score at $N = 10$. Generally, higher metric values indicate better ranking accuracy.

[4]https://grouplens.org/datasets/movielens/
[5]https://grouplens.org/datasets/hetrec-2011/.
[6]http://www.yelp.com/dataset-challenge
[7]https://www.imdb.com/.

Comparison Methods. Four types of recommendation methods are compared: **(1)** plain collaborative filtering (CF) [34] based methods that only leverage user-item interactions, including Mostpop, UserKNN [17], ItemKNN [34], BPRMF [33], NeuMF [15]; **(2)** direct-relation based method with KGs – CKE [58]; **(3)** semantic-path based methods with KGs, including FMG [59], MCRec [19], RKGE [40], KPRN [52]; and **(4)** propagation based method with KGs – KGAT [50].

Note that we do not compare with other KG-based recommendation methods (e.g., DKN [46], SHINE [43], PER [57], SemRec [37], HERec [36]), as they are generally outperformed by recently proposed methods: RKGE [40], KPRN [52] and KGAT [50].

Parameter Settings. The optimal parameter settings for all the comparison methods are achieved by either empirical study or adopting the settings as suggested by the original papers. For AKGE, we adopt Adam [23] as the optimizer, and apply a grid search in $\{0.001, 0.002, 0.01, 0.02\}$ to find out the best learning rate γ; the optimal setting for L_2 regularization coefficient λ is searched in $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$. We empirically find out the best settings for other parameters as follows: the batch size is 256; the embedding sizes of entity, entity type and relation are set as $d = 128, d' = d'' = 32$; the hidden state size is 128; and the size of predictive factors of MLP component is set to 32. In addition, the number of sampled paths K for each user-item pair is set to 30 and the propagation step $T = 2$ to avoid over-fitting [54]. The detailed analysis of parameter sensitivity will be introduced in Section 4.5.

4.2 Performance Comparison (RQ1)

Table 2 reports the performance of all the comparisons on the three datasets w.r.t. Hit@10 and NDCG@10. Due to space limit, we only show the results at $N = 10$, and similar trends can be observed at $N = \{5, 15, 20\}$. We summarize the major findings from the experimental results as below:

(1) Among all the comparison methods, most KG-based methods outperform the plain CF-based methods on the three datasets across all the evaluation metrics, which demonstrates that the usage of KGs indeed greatly improves the recommendation performance. It is worthwhile to note that NeuMF achieves better performance than BPRMF, implying the effectiveness of neural networks in capturing complex user-item interactions for recommendation;

(2) The direct-relation based method CKE underperforms the semantic path-based methods (RKGE and KPRN), indicating that modeling only first-order relations might not be able to make full use of the rich information encoded in KGs. It meanwhile confirms the effectiveness of semantic paths on recommendation. The observation is consistent with [45, 52]. In terms of semantic path-based baselines, both KPRN and RKGE are based on the automatically

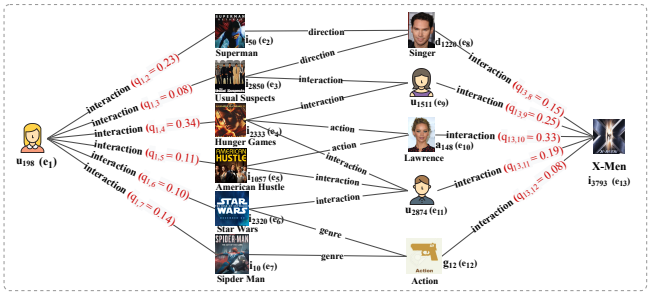


Figure 4: An illustrative example from MI-1M, where the extracted subgraph connecting the user-item pair (i.e., u_{198} – X-Men) contains users (e.g., u_{198}), movies (e.g., Superman), directors (e.g., Singer), actors (e.g., Lawrence) and genres (e.g., Action) as entities; interaction (e.g., u_{198} interacts with Superman), direction (e.g., Superman is directed by Singer), action (e.g., Hunger Game is acted by Lawrence) and genre (Star Wars belongs to action) as entity relations. $e_1, \cdots, e_{13}$ respectively denote the mapped indices of entities in the constructed subgraph; and the number in red (e.g., $q_{1,2} = 0.23$) attaching on the link is the attention weight for neighbors (e.g., e_2) w.r.t the target entity (e.g., e_1).

mined semantic paths, while FMG and MCRec heavily rely on the quality of handcrafted meta-paths. The performance of KPRN and RKGE far exceeds that of FMG and MCRec, verifying that the pre-defined features fail to uncover all potential relations between entities. Though both KPRN and RKGE are based on semantic paths, KPRN performs better than RKGE, as it additionally takes entity types and relation types into consideration;

(3) The most recent propagation based method KGAT outperforms path-based methods (KPRN and RKGE) on all datasets, which implies the effectiveness of explicitly modeling high-order connectivity. However, KGAT has a larger gap with AKGE especially on the dense dataset MI-1M, where the user preference requires to be uncovered with a large number of interactions. Propagating over the full KG could not distill the effective information from the noise, thus limiting the performance of KGAT;

(4) Overall, our proposed AKGE consistently achieves the best performance among all the comparisons on the three datasets w.r.t. all evaluation metrics. The improvements achieved by AKGE relative to the runner up w.r.t. Hit and NDCG are 8.45%, 8.65% on average, respectively (Paired t-test with p-value < 0.01). In a nutshell, through a better exploitation on both semantic and topological information encoded in KGs, AKGE shows its effectiveness for personalized recommendation.

4.3 Case Study (RQ2)

We conduct a case study to demonstrate AKGE's capability of providing potential explanations for the recommendation results. Towards this end, we randomly select a user (u_{198}) and her test item (i_{3797}) in MI-1M as an illustrative example. As shown in Figure 4, it depicts the subgraph connecting u_{198}–i_{3797}, which is automatically extracted via the Subgraph Construction Module. We also attach the attention weights (red numbers) learned from the AGNN Module for our target user (u_{198}) and target item (i_{3797}) on the links. It can be observed that our AKGE can discriminate the influence

Table 3: Effects of different pre-train techniques.

	MI-1M		Last-FM		Yelp	
	Hit@10	NDCG@10	Hit@10	NDCG@10	Hit@10	NDCG@10
AKGE$_{TransE}$	0.8871	0.6902	0.8829	0.6813	0.8591	0.6657
AKGE$_{TransH}$	0.8942	0.6951	0.8865	0.6894	0.8635	0.6731
AKGE$_{TransR}$	**0.9035**	**0.7096**	**0.8961**	**0.6987**	**0.8842**	**0.6827**

Table 4: Effects of different path sampling strategies.

	MI-1M		Last-FM		Yelp	
	Hit@10	NDCG@10	Hit@10	NDCG@10	Hit@10	NDCG@10
AKGE$_{rand}$	0.8726	0.6735	0.8624	0.6691	0.8406	0.6529
AKGE$_{mp}$	0.8815	0.6876	0.8742	0.6725	0.8568	0.6581
AKGE	**0.9034**	**0.7095**	**0.8959**	**0.6986**	**0.8841**	**0.6825**

Table 5: Effects of entity type, relation-aware propagation and attentive aggregation.

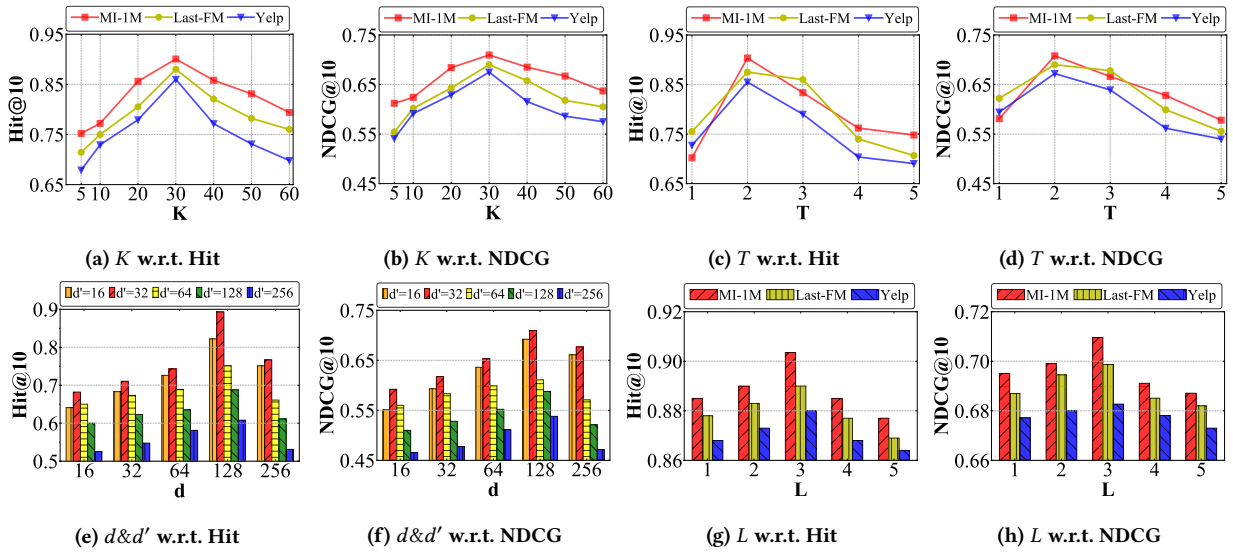
	MI-1M		Last-FM		Yelp	
	Hit@10	NDCG@10	Hit@10	NDCG@10	Hit@10	NDCG@10
AKGE$_{w/o\ type}$	**0.8951**	**0.6984**	**0.8903**	**0.6901**	**0.8785**	**0.6784**
AKGE$_{w/o\ rel}$	0.8903	0.6895	0.8837	0.6874	0.8695	0.6711
AKGE$_{w/o\ att}$	0.8814	0.6796	0.8798	0.6753	0.8482	0.6587
AKGE$_{plain}$	0.8736	0.6757	0.8684	0.6702	0.8468	0.6491

of neighbors when learning embedding for the target user via attentive aggregation, thus offering potential explanations for her preference. As we can see, i_{2333} gets the largest attention weight among all the neighbors (i.e., $i_{50}, i_{2850}, i_{2333}, i_{1057}, i_{2320}, i_{10}$) of the target user u_{198}, indicating i_{2333} plays the key role to explain her intents, that is, contributes more information for learning embedding of u_{198}. It makes sense since i_{2333} has a strong similarity with the recommended movie i_{3793}: sharing the same actor a_{148} and being watched by the same users u_{1511} and u_{2874}.

4.4 Detailed Study of AKGE (RQ3)

Effects of Pre-train Technique. To study the effects of different pre-train techniques on AKGE, we select three widely used graph embedding methods: TransE [2], TransH [53] and TransR [28]. We compare three variants of AKGE with different pre-train techniques, including AKGE$_{TransE}$, AKGE$_{TransH}$ and AKGE$_{TransR}$. The results are shown in Table 3, from which we can observe that AKGE$_{TransH}$ performs better than AKGE$_{TransE}$. One possible reason is that TransH enables an entity to have distinct embeddings when involved in different relations by using relation hyperplanes. AKGE$_{TransR}$ performs the best among all the variants. This is probably because TransR represents entities and relations in different spaces bridged by relation-specific matrices, thus is capable of better capturing the first-order relations among the KG.

Effects of Path Sampling Strategy. To evaluate the effectiveness of our proposed *distance-aware sampling strategy*, we compare AKGE with its two variants: (1) *AKGE$_{rand}$* – it utilizes the random sampling strategy to select paths for subgraph construction; (2) *AKGE$_{mp}$* – it constructs subgraph with selected meta-paths based on [59]. The results are reported in Table 4, where we notice that AKGE$_{mp}$ performs better than AKGE$_{rand}$, while both of them are outpeformed by AKGE. This can be explained as: with random sampling, AKGE$_{rand}$ may introduce some noise (e.g. remote neighbors and weak relations) into the generated paths; whereas the carefully designed meta-paths with different semantics in AKGE$_{mp}$ help improve the quality of paths to some extent; thanks to the *distance-aware sampling strategy*, AKGE is able to automatically uncover



(a) K w.r.t. Hit (b) K w.r.t. NDCG (c) T w.r.t. Hit (d) T w.r.t. NDCG

(e) $d\&d'$ w.r.t. Hit (f) $d\&d'$ w.r.t. NDCG (g) L w.r.t. Hit (h) L w.r.t. NDCG

Figure 5: Parameter sensitivity, where (a-b), (c-d), (e-f) and (g-h) respectively show the results on the number of paths, the propagation steps, the embedding sizes and the number of layers of MLP.

and discriminate the potential relations between user-item pairs. Moreover, the fact that AKGE$_{rand}$ outperforms other baselines (e.g. KPRN as shown in Table 2) illustrates the effectiveness of AKGE on modeling complex semantics within KGs.

Effects of Entity Type, Relation-aware Propagation and Attentive Aggregation. To fully exploit the heterogeneity of KGs, AGNN module takes entity type, relation-aware propagation and attentive aggregation into consideration. To study their respective effects, we compare four variants of AKGE: (1) AKGE$_{w/o\ type}$ - it removes the entity type embedding in Equation 1; (2) AKGE$_{w/o\ rel}$ - it removes the relation-aware hidden states, but keeps the original hidden states in Equation 3; (3) AKGE$_{w/o\ att}$ - it removes the attention weight vector in Equation 5; (4) AKGE$_{plain}$ - it removes the above three components. The results are reported in Table 5, where we notice that AKGE$_{plain}$ performs the worst among the four variants, which demonstrates that the three components indeed greatly improve the recommendation performance. This is further verified by the fact that all the other three variants (1,2,3) perform worse than AKGE (shown in Table 4). In addition, the performance decrease caused by removing relation is larger than that caused by removing entity type, while is smaller than that caused by removing attention. This helps validate the great benefits brought by the proposed attentive aggregation.

4.5 Parameter Sensitivity (RQ4)

We finally study how different settings of hyper-parameters affect the performance of AKGE. In the following, except for the parameter being tested, all the other parameters are fixed as introduced in Section 4.1 for a fair comparison.

Number of Sampled Paths. To investigate the impact of the number of sampled paths (K) on recommendation accuracy, we fuse different number of paths with $K = \{5, 10, 20, 30, 40, 50, 60\}$ into AKGE. As shown in Figures 5 (a-b), we observe that as K increases, the performance of AKGE greatly improves at first since more paths could help encode rich knowledge from the KG. The optimal performance is obtained with $K = 30$, while gradually drops with further increase of K, which implies that too much integration of semantic paths would introduce noise even dramatically degrade the performance (e.g. accuary and complexity) of AKGE.

Propagation Steps of AKGE. We vary the propagation steps T to study its influence on the performance of AKGE. In particular, we search T in the range of $\{1, 2, 3, 4\}$. The results are presented in Figures 5 (c-d), where we observe that AKGE achieves a better performance with $T = 2$ over $T = 1$. This suggests that increasing the depth of AKGE enables to efficiently model high-order connectivity, thus substantially enhancing the performance. However, the performance evidently decreases when stacking more steps (i.e., $T = 3$ and $T = 4$). This implies that considering second-order relations among entities could be sufficient to help capture the compositional semantics encoded in the subgraph, while stacking more steps may lead to over-fitting [47, 54].

Embedding Sizes of AKGE. We further examine how the sizes of entity embedding d and entity type embedding d' affect the performance of AKGE. To achieve this, we test all combinations of d and d' in the range of $\{16, 32, 64, 128, 256\}$. The results are shown in Figures 5 (e-f). Due to space limit, we only show the results on MI-1M, and similar trends can be observed on the other datasets. From the results, we note that, given a fixed size of entity type embedings (e.g., $d' = 16$), the performance of AKGE first improves with the increase of entity embedding size d, and the best performance is achieved when $d = 128$. This suggests embeddings with larger size can remarkably help encode useful information. The performance, however, drops a lot with further increase of d (i.e., $d = 256$), as oversized embeddings may over-represent the entity, thus introducing noise. Similar trends are also possessed by d', and its optimal value is 32 for the three datasets.

Layers of MLP. We vary the number of layers of MLP in the range of $L = \{1, 2, 3, 4, 5\}$ to study its influence on the performance of AKGE. The results are reported in Figures 5 (g-h), where we can

note a gradual enhancement on the performance as L increases at first. This helps indicate the superiority of high non-linearities brought by stacking more non-linear layers for modeling complex user-item interactions. The optimal performance is achieved with $L = 3$, while drops by degrees with further increment of layer size due to possible over-fitting [15].

5 CONCLUSIONS AND FUTURE WORK

We propose a novel attentive knowledge graph embedding (AKGE) framework to better employ KGs for effective recommendation. The distance-aware sampling strategy helps automatically mine high-order subgraphs linking user-item pairs. The novel attentive graph neural network with relation-aware propagation and attentive aggregation assists in learning better semantic embeddings for entities in subgraphs by fully considering the heterogeneity of KGs. Extensive validation shows the superiority of AKGE against other counterparts. In the future, we aim to further reduce the model complexity of AKGE.

REFERENCES

[1] Immanuel Bayer, Xiangnan He, Bhargav Kanagal, et al. 2017. A generic coordinate descent framework for learning from implicit feedback. In *WWW*.

[2] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, et al. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*.

[3] Yixin Cao, Xiang Wang, Xiangnan He, et al. 2019. Unifying Knowledge Graph Learning and Recommendation: Towards a Better Understanding of User Preferences. In *WWW*. ACM.

[4] Rose Catherine and William Cohen. 2016. Personalized recommendations using knowledge graphs: A probabilistic logic programming approach. In *RecSys*.

[5] Yves Chauvin and David E Rumelhart. 2013. *Backpropagation: theory, architectures, and applications*.

[6] Ting Chen and Yizhou Sun. 2017. Task-guided and path-augmented heterogeneous network embedding for author identification. In *WSDM*.

[7] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, et al. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[8] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *SIGKDD*.

[9] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. 2015. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *WWW*.

[10] Dumitru Erhan, Yoshua Bengio, Aaron Courville, et al. 2010. Why does unsupervised pre-training help deep learning? *JMLR* 11, Feb (2010).

[11] Wenqi Fan, Yao Ma, Qing Li, et al. 2019. Graph Neural Networks for Social Recommendation. *arXiv preprint arXiv:1902.07243* (2019).

[12] Tao-yang Fu, Wang-Chien Lee, and Zhen Lei. 2017. Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. In *CIKM*.

[13] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NIPS*.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

[15] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*.

[16] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *SIGIR*.

[17] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, et al. 1999. An Algorithmic Framework for Performing Collaborative Filtering. In *SIGIR (SIGIR '99)*. ACM, New York, NY, USA, 8. https://doi.org/10.1145/312624.312682

[18] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. 2017. Collaborative metric learning. In *WWW*.

[19] Binbin Hu, Chuan Shi, Wayne Xin Zhao, and Philip S Yu. 2018. Leveraging meta-path based context for top-n recommendation with a neural co-attention model. In *SIGKDD*.

[20] Jin Huang, Wayne Xin Zhao, Hongjian Dou, et al. 2018. Improving sequential recommendation with knowledge-enhanced memory networks. In *SIGIR*.

[21] Zhipeng Huang and Nikos Mamoulis. 2017. Heterogeneous information network embedding for meta path based proximity. *arXiv preprint arXiv:1701.05291* (2017).

[22] Guoliang Ji, Shizhu He, Liheng Xu, et al. 2015. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 687–696.

[23] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[24] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[25] Loic Landrieu and Martin Simonovsky. 2018. Large-scale point cloud semantic segmentation with superpoint graphs. In *CVPR*.

[26] Aaron Q Li, Amr Ahmed, Sujith Ravi, and Alexander J Smola. 2014. Reducing the sampling complexity of topic models. In *SIGKDD*. ACM.

[27] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2015. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493* (2015).

[28] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*.

[29] Zemin Liu, Vincent W Zheng, Zhou Zhao, Zhao Li, Hongxia Yang, Minghui Wu, and Jing Ying. 2018. Interactive paths embedding for semantic proximity search on heterogeneous graphs. In *SIGKDD*.

[30] 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 *AAAI*.

[31] Chen Luo, Wei Pang, Zhe Wang, and Chenghua Lin. 2014. Hete-cf: Social-based collaborative filtering recommendation using heterogeneous relations. In *ICDM*.

[32] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, Vol. 30.

[33] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*.

[34] Badrul Munir Sarwar, George Karypis, Joseph A Konstan, John Riedl, et al. 2001. Item-based collaborative filtering recommendation algorithms. *WWW* 1 (2001), 285–295.

[35] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The graph neural network model. *IEEE Transactions on Neural Networks* 20, 1 (2009).

[36] Chuan Shi, Binbin Hu, Wayne Xin Zhao, and S Yu Philip. 2019. Heterogeneous information network embedding for recommendation. *TKDE* 31, 2 (2019).

[37] 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 *CIKM*.

[38] Yizhou Sun and Jiawei Han. 2013. Mining heterogeneous information networks: a structural analysis approach. *ACM SIGKDD Explorations Newsletter* 14, 2 (2013).

[39] 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).

[40] Zhu Sun, Jie Yang, Jie Zhang, Alessandro Bozzon, Long-Kai Huang, and Chi Xu. 2018. Recurrent knowledge graph embedding for effective recommendation. In *RecSys*. 297–305.

[41] Robert Tarjan. 1972. Depth-first search and linear graph algorithms. *SIAM J. Comput.* 1, 2 (1972), 146–160.

[42] Damien Teney, Lingqiao Liu, and Anton van den Hengel. 2017. Graph-structured representations for visual question answering. In *CVPR*.

[43] Hongwei Wang, Fuzheng Zhang, Min Hou, Xing Xie, Minyi Guo, and Qi Liu. 2018. Shine: Signed heterogeneous information network embedding for sentiment link prediction. In *WSDM*. 592–600.

[44] Hongwei Wang, Fuzheng Zhang, Jialin Wang, et al. 2019. Exploring High-Order User Preference on the Knowledge Graph for Recommender Systems. *TOIS* 37, 3 (2019).

[45] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *CIKM*.

[46] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. Dkn: Deep knowledge-aware network for news recommendation. In *WWW*.

[47] Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. 2019. Knowledge-aware Graph Neural Networks with Label Smoothness Regularization for Recommender Systems. In *KDD*.

[48] Hongwei Wang, Fuzheng Zhang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2019. Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation. In *WWW*. ACM.

[49] Hongwei Wang, Miao Zhao, Xing Xie, et al. 2019. Knowledge graph convolutional networks for recommender systems. In *WWW*. ACM.

[50] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. *KDD* (2019).

[51] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Peng Cui, P. Yu, and Yanfang Ye. 2019. Heterogeneous Graph Attention Network. In *WWW*.

[52] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2018. Explainable Reasoning over Knowledge Graphs for Recommendation. *AAAI* (2018).

[53] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*.

[54] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2018. Session-based Recommendation with Graph Neural Networks. *AAAI* (2018).
[55] Xin Xin, Xiangnan He, Yongfeng Zhang, et al. 2019. Relational Collaborative Filtering: Modeling Multiple Item Relations for Recommendation. In *SIGIR*.
[56] Bishan Yang, Wen-tau Yih, Xiaodong He, et al. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575* (2014).
[57] 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 *WSDM*. 283–292.
[58] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *SIGKDD*.
[59] Huan Zhao, Quanming Yao, Jianda Li, Yangqiu Song, and Dik Lun Lee. 2017. Meta-graph based recommendation fusion over heterogeneous information networks. In *SIGKDD*. 635–644.
[60] Rong Zhou and Eric A Hansen. 2006. Breadth-first heuristic search. *Artificial Intelligence* 170, 4-5 (2006).