KGAT: Knowledge Graph Attention Network for Recommendation

Xiang Wang
National University of Singapore
xiangwang@u.nus.edu

Xiangnan He*
University of Science and Technology of China
xiangnanhe@gmail.com

Yixin Cao
National University of Singapore
caoyixin2011@gmail.com

Meng Liu
Shandong University
mengliu.sdu@gmail.com

Tat-Seng Chua
National University of Singapore
dcscts@nus.edu.sg

ABSTRACT

To provide more accurate, diverse, and explainable recommendation, it is compulsory to go beyond modeling user-item interactions and take side information into account. Traditional methods like factorization machine (FM) cast it as a supervised learning problem, which assumes each interaction as an independent instance with side information encoded. Due to the overlook of the relations among instances or items (*e.g.,* the director of a movie is also an actor of another movie), these methods are insufficient to distill the collaborative signal from the collective behaviors of users.

In this work, we investigate the utility of knowledge graph (KG), which breaks down the independent interaction assumption by linking items with their attributes. We argue that in such a hybrid structure of KG and user-item graph, high-order relations — which connect two items with one or multiple linked attributes — are an essential factor for successful recommendation. We propose a new method named *Knowledge Graph Attention Network* (KGAT) which explicitly models the high-order connectivities in KG in an end-to-end fashion. It recursively propagates the embeddings from a node's neighbors (which can be users, items, or attributes) to refine the node's embedding, and employs an attention mechanism to discriminate the importance of the neighbors. Our KGAT is conceptually advantageous to existing KG-based recommendation methods, which either exploit high-order relations by extracting paths or implicitly modeling them with regularization. Empirical results on three public benchmarks show that KGAT significantly outperforms state-of-the-art methods like Neural FM [11] and RippleNet [29]. Further studies verify the efficacy of embedding propagation for high-order relation modeling and the interpretability benefits brought by the attention mechanism. We release the codes and datasets at https://github.com/xiangwang1223/knowledge_graph_attention_network.

*Xiangnan He is the corresponding author.

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330989

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Collaborative Filtering, Recommendation, Graph Neural Network, Higher-order Connectivity, Embedding Propagation, Knowledge Graph

ACM Reference Format:
Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330989

1 INTRODUCTION

The success of recommendation system makes it prevalent in Web applications, ranging from search engines, E-commerce, to social media sites and news portals — without exaggeration, almost every service that provides content to users is equipped with a recommendation system. To predict user preference from the key (and widely available) source of user behavior data, much research effort has been devoted to collaborative filtering (CF) [12, 13, 32]. Despite its effectiveness and universality, CF methods suffer from the inability of modeling side information [30, 31], such as item attributes, user profiles, and contexts, thus perform poorly in sparse situations where users and items have few interactions. To integrate such information, a common paradigm is to transform them into a generic feature vector, together with user ID and item ID, and feed them into a supervised learning (SL) model to predict the score. Such a SL paradigm for recommendation has been widely deployed in industry [7, 24, 40], and some representative models include factorization machine (FM) [23], NFM (neural FM) [11], Wide&Deep [7], and xDeepFM [18], *etc.*

Although these methods have provided strong performance, a deficiency is that they model each interaction as an independent data instance and do not consider their relations. This makes them insufficient to distill attribute-based collaborative signal from the collective behaviors of users. As shown in Figure 1, there is an interaction between user u_1 and movie i_1, which is *directed by* the person e_1. CF methods focus on the histories of similar users who also watched i_1, *i.e.,* u_4 and u_5; while SL methods emphasize the similar items with the attribute e_1, *i.e.,* i_2. Obviously, these two types of information not only are complementary for recommendation,

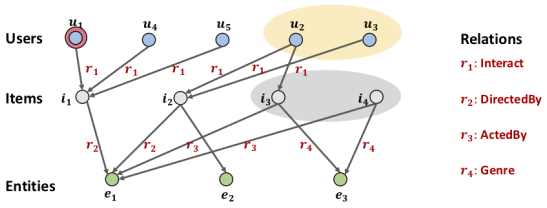


Figure 1: A toy example of collaborative knowledge graph. u_1 is the target user to provide recommendation for. The yellow circle and grey circle denote the important users and items discovered by high-order relations but are overlooked by traditional methods. Best view in color.

but also form a high-order relationship between a target user and item together. However, existing SL methods fail to unify them and cannot take into account the high-order connectivity, such as the users in the yellow circle who watched **other movies** directed by the same person e_1, or the items in the grey circle that share **other common relations** with e_1.

To address the limitation of feature-based SL models, a solution is to take the graph of item side information, *aka.* knowledge graph[1] [3, 4], into account to construct the predictive model. We term the hybrid structure of knowledge graph and user-item graph as *collaborative knowledge graph* (CKG). As illustrated in Figure 1, the key to successful recommendation is to fully exploit the high-order relations in CKG, *e.g.,* the long-range connectivities:

- $u_1 \xrightarrow{r_1} i_1 \xrightarrow{-r_2} e_1 \xrightarrow{r_2} i_2 \xrightarrow{-r_1} \{u_2, u_3\}$,
- $u_1 \xrightarrow{r_1} i_1 \xrightarrow{-r_2} e_1 \xrightarrow{r_3} \{i_3, i_4\}$,

which represent the way to the yellow and grey circle, respectively. Nevertheless, to exploit such high-order information the challenges are non-negligible: 1) the nodes that have high-order relations with the target user increase dramatically with the order size, which imposes computational overload to the model, and 2) the high-order relations contribute unequally to a prediction, which requires the model to carefully weight (or select) them.

Several recent efforts have attempted to leverage the CKG structure for recommendation, which can be roughly categorized into two types, path-based [14, 25, 29, 33, 37, 39] and regularization-based [5, 15, 33, 38]:

- Path-based methods extract paths that carry the high-order information and feed them into predictive model. To handle the large number of paths between two nodes, they have either applied path selection algorithm to select prominent paths [25, 33], or defined meta-path patterns to constrain the paths [14, 36]. One issue with such two-stage methods is that the first stage of path selection has a large impact on the final performance, but it is not optimized for the recommendation objective. Moreover, defining effective meta-paths requires domain knowledge, which can be rather labor-intensive for complicated KG with diverse types of relations and entities, since many meta-paths have to be defined to retain model fidelity.
- Regularization-based methods devise additional loss terms that capture the KG structure to regularize the recommender model learning. For example, KTUP [5] and CFKG [1] jointly train the two tasks of recommendation and KG completion with shared item embeddings. Instead of directly plugging high-order relations into the model optimized for recommendation, these methods only encode them in an implicit manner. Due to the lack of an explicit modeling, neither the long-range connectivities are guaranteed to be captured, nor the results of high-order modeling are interpretable.

Considering the limitations of existing solutions, we believe it is of critical importance to develop a model that can exploit high-order information in KG in an efficient, explicit, and end-to-end manner. Towards this end, we take inspiration from the recent developments of graph neural networks [9, 17, 28], which have the potential of achieving the goal but have not been explored much for KG-based recommendation. Specifically, we propose a new method named *Knowledge Graph Attention Network* (KGAT), which is equipped with two designs to correspondingly address the challenges in high-order relation modeling: 1) recursive embedding propagation, which updates a node's embedding based on the embeddings of its neighbors, and recursively performs such embedding propagation to capture high-order connectivities in a linear time complexity; and 2) attention-based aggregation, which employs the neural attention mechanism [6, 27] to learn the weight of each neighbor during a propagation, such that the attention weights of cascaded propagations can reveal the importance of a high-order connectivity. Our KGAT is conceptually advantageous to existing methods in that: 1) compared with path-based methods, it avoids the labor-intensive process of materializing paths, thus is more efficient and convenient to use, and 2) compared with regularization-based methods, it directly factors high-order relations into the predictive model, thus all related parameters are tailored for optimizing the recommendation objective.

The contributions of this work are summarized as follows:

- We highlight the importance of explicitly modeling the high-order relations in collaborative knowledge graph to provide better recommendation with item side information.
- We develop a new method KGAT, which achieves high-order relation modeling in an explicit and end-to-end manner under the graph neural network framework.
- We conduct extensive experiments on three public benchmarks, demonstrating the effectiveness of KGAT and its interpretability in understanding the importance of high-order relations.

2 TASK FORMULATION

We first introduce the concept of CKG and highlight the high-order connectivity among nodes, as well as the compositional relations.

User-Item Bipartite Graph: In a recommendation scenario, we typically have historical user-item interactions (*e.g.,* purchases and clicks). Here we represent interaction data as a user-item bipartite graph $\mathcal{G}_1$, which is defined as $\{(u, y_{ui}, i)|u \in \mathcal{U}, i \in \mathcal{I}\}$, where $\mathcal{U}$ and $\mathcal{I}$ separately denote the user and item sets, and a link $y_{ui} = 1$ indicates that there is an observed interaction between user u and item i; otherwise $y_{ui} = 0$.

Knowledge Graph. In addition to the interactions, we have side information for items (*e.g.,* item attributes and external knowledge). Typically, such auxiliary data consists of real-world entities and

[1] A KG is typically described as a heterogeneous network consisting of entity-relation-entity triplets, where the entity can be an item or an attribute.

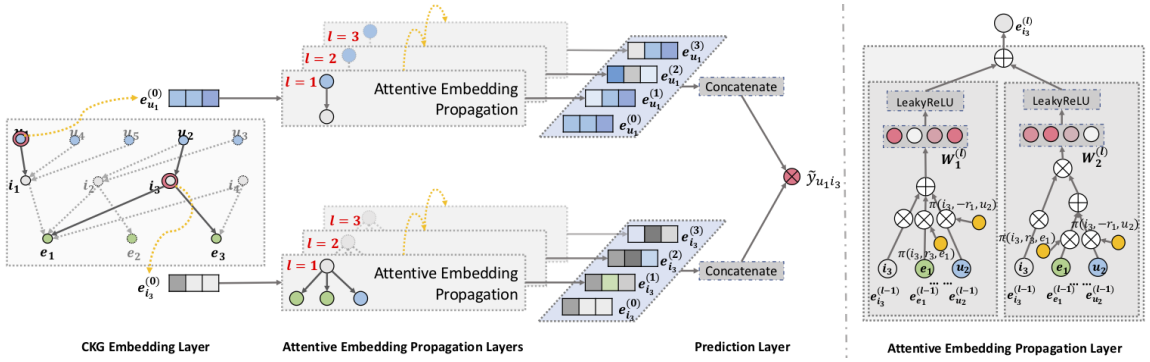


Figure 2: Illustration of the proposed KGAT model. The left subfigure shows model framework of KGAT, and the right subfigure presents the attentive embedding propagation layer of KGAT.

relationships among them to profile an item. For example, a movie can be described by its director, cast, and genres. We organize the side information in the form of knowledge graph $\mathcal{G}_2$, which is a directed graph composed of subject-property-object triple facts [5]. Formally, it is presented as $\{(h, r, t)|h, t \in \mathcal{E}, r \in \mathcal{R}\}$, where each triplet describes that there is a relationship r from head entity h to tail entity t. For example, (*Hugh Jackman, ActorOf, Logan*) states the fact that Hugh Jackman is an actor of the movie Logan. Note that $\mathcal{R}$ contains relations in both canonical direction (*e.g., ActorOf*) and inverse direction (*e.g., ActedBy*). Moreover, we establish a set of item-entity alignments $\mathcal{A} = \{(i, e)|i \in \mathcal{I}, e \in \mathcal{E}\}$, where (i, e) indicates that item i can be aligned with an entity e in the KG.

Collaborative Knowledge Graph. Here we define the concept of CKG, which encodes user behaviors and item knowledge as a unified relational graph. We first represent each user behavior as a triplet, $(u, \textit{Interact}, i)$, where $y_{ui} = 1$ is represented as an additional relation *Interact* between user u and item i. Then based on the item-entity alignment set, the user-item graph can be seamlessly integrated with KG as a unified graph $\mathcal{G} = \{(h, r, t)|h, t \in \mathcal{E}', r \in \mathcal{R}'\}$, where $\mathcal{E}' = \mathcal{E} \cup \mathcal{U}$ and $\mathcal{R}' = \mathcal{R} \cup \{\textit{Interact}\}$.

Task Description We now formulate the recommendation task to be addressed in this paper:

- **Input**: collaborative knowledge graph $\mathcal{G}$ that includes the user-item bipartite graph $\mathcal{G}_1$ and knowledge graph $\mathcal{G}_2$.
- **Output**: a prediction function that predicts the probability $\hat{y}_{ui}$ that user u would adopt item i.

High-Order Connectivity. Exploiting high-order connectivity is of importance to perform high-quality recommendation. Formally, we define the *L-order connectivity* between nodes as a multi-hop relation path: $e_0 \xrightarrow{r_1} e_1 \xrightarrow{r_2} \cdots \xrightarrow{r_L} e_L$, where $e_l \in \mathcal{E}'$ and $r_l \in \mathcal{R}'$; (e_{l-1}, r_l, e_l) is the l-th triplet, and L is the length of the sequence. To infer user preference, CF methods build upon behavior similarity among users — more specifically similar users would exhibit similar preferences on items. Such intuition can be represented as behavior-based connectivity like $u_1 \xrightarrow{r_1} i_1 \xrightarrow{-r_1} u_2 \xrightarrow{r_1} i_2$, which suggests that u_1 would exhibit preference on i_2, since her similar user u_2 has adopted i_2 before. Distinct from CF methods, SL models like FM and NFM focus on attributed-based connectivity, assuming that users tend to adopt items that share similar properties. For example, $u_1 \xrightarrow{r_1} i_1 \xrightarrow{r_2} e_1 \xrightarrow{-r_2} i_2$ suggests that u_1 would adopt i_2 since it has the same director e_1 with i_1 she liked before. However, FM and NFM treat entities as the values of individual feature fields, failing to reveal relatedness across fields and related instances. For instance, it is hard to model $u_1 \xrightarrow{r_1} i_1 \xrightarrow{r_2} e_1 \xrightarrow{-r_3} i_2$, although e_1 serves as the bridge connecting *director* and *actor* fields. We therefore argue that these methods do not fully explore the high-order connectivity and leave compositional high-order relations untouched.

3 METHODOLOGY

We now present the proposed KGAT model, which exploits high-order relations in an end-to-end fashion. Figure 2 shows the model framework, which consists of three main components: 1) embedding layer, which parameterizes each node as a vector by preserving the structure of CKG; 2) attentive embedding propagation layers, which recursively propagate embeddings from a node's neighbors to update its representation, and employ knowledge-aware attention mechanism to learn the weight of each neighbor during a propagation; and 3) prediction layer, which aggregates the representations of a user and an item from all propagation layers, and outputs the predicted matching score.

3.1 Embedding Layer

Knowledge graph embedding is an effective way to parameterize entities and relations as vector representations, while preserving the graph structure. Here we employ TransR [19], a widely used method, on CKG. To be more specific, it learns embeds each entity and relation by optimizing the translation principle $\mathbf{e}_h^r + \mathbf{e}_r \approx \mathbf{e}_t^r$, if a triplet (h, r, t) exists in the graph. Herein, $\mathbf{e}_h, \mathbf{e}_t \in \mathbb{R}^d$ and $\mathbf{e}_r \in \mathbb{R}^k$ are the embedding for h, t, and r, respectively; and $\mathbf{e}_h^r, \mathbf{e}_t^r$ are the projected representations of $\mathbf{e}_h$ and $\mathbf{e}_t$ in the relation r's space. Hence, for a given triplet (h, r, t), its plausibility score (or energy score) is formulated as follows:

$$g(h, r, t) = \|\mathbf{W}_r\mathbf{e}_h + \mathbf{e}_r - \mathbf{W}_r\mathbf{e}_t\|_2^2, \tag{1}$$

where $\mathbf{W}_r \in \mathbb{R}^{k \times d}$ is the transformation matrix of relation r, which projects entities from the d-dimension entity space into the k-dimension relation space. A lower score of $g(h, r, t)$ suggests that the triplet is more likely to be true true, and vice versa.

The training of TransR considers the relative order between valid triplets and broken ones, and encourages their discrimination

through a pairwise ranking loss:

$$\mathcal{L}_{\text{KG}} = \sum_{(h,r,t,t')\in\mathcal{T}} -\ln\sigma\Big(g(h,r,t') - g(h,r,t)\Big), \tag{2}$$

where $\mathcal{T} = \{(h,r,t,t')|(h,r,t)\in\mathcal{G}, (h,r,t')\notin\mathcal{G}\}$, and (h,r,t') is a broken triplet constructed by replacing one entity in a valid triplet randomly; $\sigma(\cdot)$ is the sigmoid function. This layer models the entities and relations on the granularity of triples, working as a regularizer and injecting the direct connections into representations, and thus increases the model representation ability (evidences in Section 4.4.3.)

3.2 Attentive Embedding Propagation Layers

Next we build upon the architecture of graph convolution network [17] to recursively propagate embeddings along high-order connectivity; moreover, by exploiting the idea of graph attention network [28], we generate attentive weights of cascaded propagations to reveal the importance of such connectivity. Here we start by describing a single layer, which consists of three components: *information propagation*, *knowledge-aware attention*, and *information aggregation*, and then discuss how to generalize it to multiple layers.

Information Propagation: One entity can be involved in multiple triplets, serving as the bridge connecting two triplets and propagating information. Taking $e_1 \xrightarrow{r_2} i_2 \xrightarrow{-r_1} u_2$ and $e_2 \xrightarrow{r_3} i_2 \xrightarrow{-r_1} u_2$ as an example, item i_2 takes attributes e_1 and e_2 as inputs to enrich its own features, and then contributes user u_2's preferences, which can be simulated by propagating information from e_1 to u_2. We build upon this intuition to perform information propagation between an entity and its neighbors.

Considering an entity h, we use $\mathcal{N}_h = \{(h,r,t)|(h,r,t)\in\mathcal{G}\}$ to denote the set of triplets where h is the head entity, termed ego-network [21]. To characterize the first-order connectivity structure of entity h, we compute the linear combination of h's ego-network:

$$\mathbf{e}_{\mathcal{N}_h} = \sum_{(h,r,t)\in\mathcal{N}_h} \pi(h,r,t)\mathbf{e}_t, \tag{3}$$

where $\pi(h,r,t)$ controls the decay factor on each propagation on edge (h,r,t), indicating how much information being propagated from t to h conditioned to relation r.

Knowledge-aware Attention: We implement $\pi(h,r,t)$ via relational attention mechanism, which is formulated as follows:

$$\pi(h,r,t) = (\mathbf{W}_r\mathbf{e}_t)^\top \tanh\Big((\mathbf{W}_r\mathbf{e}_h + \mathbf{e}_r)\Big), \tag{4}$$

where we select tanh [28] as the nonlinear activation function. This makes the attention score dependent on the distance between e_h and e_t in the relation r's space, *e.g.,* propagating more information for closer entities. Note that, we employ only inner product on these representations for simplicity, and leave the further exploration of the attention module as the future work.

Hereafter, we normalize the coefficients across all triplets connected with h by adopting the softmax function:

$$\pi(h,r,t) = \frac{\exp(\pi(h,r,t))}{\sum_{(h,r',t')\in\mathcal{N}_h}\exp(\pi(h,r',t'))}. \tag{5}$$

As a result, the final attention score is capable of suggesting which neighbor nodes should be given more attention to capture collaborative signals. When performing propagation forward, the attention flow suggests parts of the data to focus on, which can be treated as explanations behind the recommendation.

Distinct from the information propagation in GCN [17] and GraphSage [9] which set the discount factor between two nodes as $1/\sqrt{|\mathcal{N}_h||\mathcal{N}_t|}$ or $1/|\mathcal{N}_t|$, our model not only exploits the proximity structure of graph, but also specify varying importance of neighbors. Moreover, distinct from graph attention network [28] which only takes node representations as inputs, we model the relation e_r between e_h and e_t, encoding more information during propagation. We perform experiments to verify the effectiveness of the attention mechanism and visualize the attention flow in Section 4.4.3 and Section 4.5, respectively.

Information Aggregation: The final phase is to aggregate the entity representation $\mathbf{e}_h$ and its ego-network representations $\mathbf{e}_{\mathcal{N}_h}$ as the new representation of entity h — more formally, $\mathbf{e}_h^{(1)} = f(\mathbf{e}_h, \mathbf{e}_{\mathcal{N}_h})$. We implement $f(\cdot)$ using the following three types of aggregators:

- *GCN Aggregator* [17] sums two representations up and applies a nonlinear transformation, as follows:

$$f_{\text{GCN}} = \text{LeakyReLU}\Big(\mathbf{W}(\mathbf{e}_h + \mathbf{e}_{\mathcal{N}_h})\Big), \tag{6}$$

  where we set the activation function set as LeakyReLU [20]; $\mathbf{W}\in\mathbb{R}^{d'\times d}$ are the trainable weight matrices to distill useful information for propagation, and d' is the transformation size.
- *GraphSage Aggregator* [9] concatenates two representations, followed by a nonlinear transformation:

$$f_{\text{GraphSage}} = \text{LeakyReLU}\Big(\mathbf{W}(\mathbf{e}_h||\mathbf{e}_{\mathcal{N}_h})\Big), \tag{7}$$

  where $||$ is the concatenation operation.
- *Bi-Interaction Aggregator* is carefully designed by us to consider two kinds of feature interactions between $\mathbf{e}_h$ and $\mathbf{e}_{\mathcal{N}_h}$, as follows:

$$\begin{aligned} f_{\text{Bi-Interaction}} =& \text{LeakyReLU}\Big(\mathbf{W}_1(\mathbf{e}_h + \mathbf{e}_{\mathcal{N}_h})\Big) + \\ & \text{LeakyReLU}\Big(\mathbf{W}_2(\mathbf{e}_h \odot \mathbf{e}_{\mathcal{N}_h})\Big), \end{aligned} \tag{8}$$

  where $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{d'\times d}$ are the trainable weight matrices, and $\odot$ denotes the element-wise product. Distinct from GCN and GraphSage aggregators, we additionally encode the feature interaction between $\mathbf{e}_h$ and $\mathbf{e}_{\mathcal{N}_h}$. This term makes the information being propagated sensitive to the affinity between $\mathbf{e}_h$ and $\mathbf{e}_{\mathcal{N}_h}$, *e.g.,* passing more messages from similar entities.

To summarize, the advantage of the embedding propagation layer lies in explicitly exploiting the first-order connectivity information to relate user, item, and knowledge entity representations. We empirically compare the three aggregators in Section 4.4.2.

High-order Propagation: We can further stack more propagation layers to explore the high-order connectivity information, gathering the information propagated from the higher-hop neighbors. More formally, in the l-th steps, we recursively formulate the

representation of an entity as:

$$\mathbf{e}_h^{(l)} = f\left(\mathbf{e}_h^{(l-1)}, \mathbf{e}_{\mathcal{N}_h}^{(l-1)}\right), \tag{9}$$

wherein the information propagated within l-ego network for the entity h is defined as follows,

$$\mathbf{e}_{\mathcal{N}_h}^{(l-1)} = \sum_{(h,r,t)\in\mathcal{N}_h} \pi(h,r,t)\mathbf{e}_t^{(l-1)}, \tag{10}$$

$\mathbf{e}_t^{(l-1)}$ is the representation of entity t generated from the previous information propagation steps, memorizing the information from its $(l-1)$-hop neighbors; $\mathbf{e}_h^{(0)}$ is set as $\mathbf{e}_h$ at the initial information-propagation iteration. It further contributes to the representation of entity h at layer l. As a result, high-order connectivity like $u_2 \xrightarrow{r_1} i_2 \xrightarrow{-r_2} e_1 \xrightarrow{r_2} i_1 \xrightarrow{-r_1} u_1$ can be captured in the embedding propagation process. Furthermore, the information from u_2 is explicitly encoded in $\mathbf{e}_{u_1}^{(3)}$. Clearly, the high-order embedding propagation seamlessly injects the attribute-based collaborative signal into the representation learning process.

3.3 Model Prediction

After performing L layers, we obtain multiple representations for user node u, namely $\{\mathbf{e}_u^{(1)}, \cdots, \mathbf{e}_u^{(L)}\}$; analogous to item node i, $\{\mathbf{e}_i^{(1)}, \cdots, \mathbf{e}_i^{(L)}\}$ are obtained. As the output of the l-th layer is the message aggregation of the tree structure depth of l rooted at u (or i) as shown in Figure 1, the outputs in different layers emphasize the connectivity information of different orders. We hence adopt the layer-aggregation mechanism [34] to concatenate the representations at each step into a single vector, as follows:

$$\mathbf{e}_u^* = \mathbf{e}_u^{(0)}\| \cdots \|\mathbf{e}_u^{(L)}, \quad \mathbf{e}_i^* = \mathbf{e}_i^{(0)}\| \cdots \|\mathbf{e}_i^{(L)}, \tag{11}$$

where $\|$ is the concatenation operation. By doing so, we not only enrich the initial embeddings by performing the embedding propagation operations, but also allow controlling the strength of propagation by adjusting L.

Finally, we conduct inner product of user and item representations, so as to predict their matching score:

$$\hat{y}(u,i) = {\mathbf{e}_u^*}^\top \mathbf{e}_i^*. \tag{12}$$

3.4 Optimization

To optimize the recommendation model, we opt for the BPR loss [22]. Specifically, it assumes that the observed interactions, which indicate more user preferences, should be assigned higher prediction values than unobserved ones:

$$\mathcal{L}_{\text{CF}} = \sum_{(u,i,j)\in O} -\ln \sigma\left(\hat{y}(u,i) - \hat{y}(u,j)\right) \tag{13}$$

where $O = \{(u,i,j)|(u,i)\in\mathcal{R}^+, (u,j)\in\mathcal{R}^-\}$ denotes the training set, $\mathcal{R}^+$ indicates the observed (positive) interactions between user u and item j while $\mathcal{R}^-$ is the sampled unobserved (negative) interaction set; $\sigma(\cdot)$ is the sigmoid function.

Finally, we have the objective function to learn Equations (2) and (13) jointly, as follows:

$$\mathcal{L}_{\text{KGAT}} = \mathcal{L}_{\text{KG}} + \mathcal{L}_{\text{CF}} + \lambda \|\Theta\|_2^2, \tag{14}$$

where $\Theta = \{\mathbf{E}, \mathbf{W}_r, \forall l\in\mathcal{R}, \mathbf{W}_1^{(l)}, \mathbf{W}_2^{(l)}, \forall l \in \{1,\cdots,L\}\}$ is the model parameter set, and $\mathbf{E}$ is the embedding table for all entities and relations; L_2 regularization parameterized by λ on Θ is conducted to prevent overfitting. It is worth pointing out that in terms of model size, the majority of model parameters comes from the entity embeddings (*e.g.,* 6.5 million on experimented Amazon dataset), which is almost identical to that of FM; the propagation layer weights are lightweight (*e.g.,* 5.4 thousand for the tower structure of three layers, *i.e.,* $64-32-16-8$, on the Amazon dataset).

3.4.1 ***Training:*** We optimize $\mathcal{L}_{KG}$ and $\mathcal{L}_{CF}$ alternatively, where mini-batch Adam [16] is adopted to optimize the embedding loss and the prediction loss. Adam is a widely used optimizer, which is able to adaptively control the learning rate *w.r.t.* the absolute value of gradient. In particular, for a batch of randomly sampled (h,r,t,t'), we update the embeddings for all nodes; hereafter, we sample a batch of (u,i,j) randomly, retrieve their representations after L steps of propagation, and then update model parameters by using the gradients of the prediction loss.

3.4.2 ***Time Complexity Analysis:*** As we adopt the alternative optimization strategy, the time cost mainly comes from two parts. For the knowledge graph embedding (*cf.* Equation (2)), the translation principle has computational complexity $O(|\mathcal{G}_2|d^2)$. For the attention embedding propagation part, the matrix multiplication of the l-th layer has computational complexity $O(|\mathcal{G}|d_l d_{l-1})$; and d_l and d_{l-1} are the current and previous transformation size. For the final prediction layer, only the inner product is conducted, for which the time cost of the whole training epoch is $O(\sum_{l=1}^{L}|\mathcal{G}|d_l)$. Finally, the overall training complexity of KGAT is $O(|\mathcal{G}_2|d^2 + \sum_{l=1}^{L}|\mathcal{G}|d_l d_{l-1} + |\mathcal{G}|d_l)$.

As online services usually require real-time recommendation, the computational cost during inference is more important that that of training phase. Empirically, FM, NFM, CFKG, CKE, GC-MC, **KGAT**, MCRec, and RippleNet cost around 700s, 780s, 800s, 420s, 500s, **560**s, 20 hours, and 2 hours for all testing instances on Amazon-Book dataset, respectively. As we can see, KGAT achieves comparable computation complexity to SL models (FM and NFM) and regularization-based methods (CFKG and CKE), being much efficient that path-based methods (MCRec and RippleNet).

4 EXPERIMENTS

We evaluate our proposed method, especially the embedding propagation layer, on three real-world datasets. We aim to answer the following research questions:

- **RQ1**: How does KGAT perform compared with state-of-the-art knowledge-aware recommendation methods?
- **RQ2**: How do different components (*i.e.,* knowledge graph embedding, attention mechanism, and aggregator selection) affect KGAT?
- **RQ3**: Can KGAT provide reasonable explanations about user preferences towards items?

4.1 Dataset Description

To evaluate the effectiveness of KGAT, we utilize three benchmark datasets: Amazon-book, Last-FM, and Yelp2018, which are publicly accessible and vary in terms of domain, size, and sparsity.

Table 1: Statistics of the datasets.

		Amazon-book	Last-FM	Yelp2018
User-Item Interaction	#Users	70,679	23,566	45,919
	#Items	24,915	48,123	45,538
	#Interactions	847,733	3,034,796	1,185,068
Knowledge Graph	#Entities	88,572	58,266	90,961
	#Relations	39	9	42
	#Triplets	2,557,746	464,567	1,853,704

Amazon-book[2]**:** Amazon-review is a widely used dataset for product recommendation [10]. We select Amazon-book from this collection. To ensure the quality of the dataset, we use the 10-core setting, *i.e.,* retaining users and items with at least ten interactions.

Last-FM[3]**:** This is the music listening dataset collected from Last.fm online music systems. Wherein, the tracks are viewed as the items. In particular, we take the subset of the dataset where the timestamp is from Jan, 2015 to June, 2015. We use the same 10-core setting in order to ensure data quality.

Yelp2018[4]**:** This dataset is adopted from the 2018 edition of the Yelp challenge. Here we view the local businesses like restaurants and bars as the items. Similarly, we use the 10-core setting to ensure that each user and item have at least ten interactions.

Besides the user-item interactions, we need to construct item knowledge for each dataset. For Amazon-book and Last-FM, we map items into Freebase entities via title matching if there is a mapping available. In particular, we consider the triplets that are directly related to the entities aligned with items, no matter which role (*i.e.,* subject or object) it serves as. Distinct from existing knowledge-aware datasets that provide only one-hop entities of items, we also take the triplets that involve two-hop neighbor entities of items into consideration. For Yelp2018, we extract item knowledge from the local business information network (*e.g.,* category, location, and attribute) as KG data. To ensure the KG quality, we then preprocess the three KG parts by filtering out infrequent entities (*i.e.,* lowever than 10 in both datasets) and retaining the relations appearing in at least 50 triplets. We summarize the statistics of three datasets in Table 1 and publish our datasets at https://github.com/xiangwang1223/knowledge_graph_attention_network.

For each dataset, we randomly select 80% of interaction history of each user to constitute the training set, and treat the remaining as the test set. From the training set, we randomly select 10% of interactions as validation set to tune hyper-parameters. For each observed user-item interaction, we treat it as a positive instance, and then conduct the negative sampling strategy to pair it with one negative item that the user did not consume before.

4.2 Experimental Settings

4.2.1 ***Evaluation Metrics.*** For each user in the test set, we treat all the items that the user has not interacted with as the negative items. Then each method outputs the user's preference scores over all the items, except the positive ones in the training set. To evaluate the effectiveness of top-K recommendation and preference ranking, we adopt two widely-used evaluation protocols [13, 35]: recall@K and ndcg@K. By default, we set $K = 20$. We report the average metrics for all users in the test set.

4.2.2 ***Baselines.*** To demonstrate the effectiveness, we compare our proposed KGAT with SL (FM and NFM), regularization-based (CFKG and CKE), path-based (MCRec and RippleNet), and graph neural network-based (GC-MC) methods, as follows:

- **FM** [23]: This is a bechmark factorization model, where considers the second-order feature interactions between inputs. Here we treat IDs of a user, an item, and its knowledge (*i.e.,* entities connected to it) as input features.
- **NFM** [11]: The method is a state-of-the-art factorization model, which subsumes FM under neural network. Specially, we employed one hidden layer on input features as suggested in [11].
- **CKE** [38]: This is a representative regularization-based method, which exploits semantic embeddings derived from TransR [19] to enhance matrix factorization [22].
- **CFKG** [1]: The model applies TransE [2] on the unified graph including users, items, entities, and relations, casting the recommendation task as the plausibility prediction of $(u, \textit{Interact}, i)$ triplets.
- **MCRec** [14]: This is a path-based model, which extracts qualified meta-paths as connectivity between a user and an item.
- **RippleNet** [29]: Such model combines regularization- and path-based methods, which enrich user representations by adding that of items within paths rooted at each user.
- **GC-MC** [26]: Such model is designed to employ GCN [17] encoder on graph-structured data, especially for the user-item bipartite graph. Here we apply it on the user-item knowledge graph. Especially, we employ one graph convolution layers as suggested in [26], where the hidden dimension is set equal to the embedding size.

4.2.3 ***Parameter Settings.*** We implement our KGAT model in Tensorflow. The embedding size is fixed to 64 for all models, except RippleNet 16 due to its high computational cost. We optimize all models with Adam optimizer, where the batch size is fixed at 1024. The default Xavier initializer [8] to initialize the model parameters. We apply a grid search for hyper-parameters: the learning rate is tuned amongst $\{0.05, 0.01, 0.005, 0.001\}$, the coefficient of L_2 normalization is searched in $\{10^{-5}, 10^{-4}, \cdots, 10^{1}, 10^{2}\}$, and the dropout ratio is tuned in $\{0.0, 0.1, \cdots, 0.8\}$ for NFM, GC-MC, and KGAT. Besides, we employ the node dropout technique for GC-MC and KGAT, where the ratio is searched in $\{0.0, 0.1, \cdots, 0.8\}$. For MCRec, we manually define several types of user-item-attribute-item meta-paths, such as *user-book-author-user* and *user-book-genre-user* for Amazon-book dataset; we set the hidden layers as suggested in [14], which is a tower structure with 512, 256, 128, 64 dimensions. For RippleNet, we set the number of hops and the memory size as 2 and 8, respectively. Moreover, early stopping strategy is performed, *i.e.,* premature stopping if recall@20 on the validation set does not increase for 50 successive epochs. To model the third-order connectivity, we set the depth of KGAT L as three with hidden dimension 64, 32, and 16, respectively; we also report the effect of layer depth in Section 4.4.1. For each layer, we conduct the Bi-Interaction aggregator.

[2]http://jmcauley.ucsd.edu/data/amazon.
[3]https://grouplens.org/datasets/hetrec-2011/.
[4]https://www.yelp.com/dataset/challenge.

Table 2: Overall Performance Comparison.

	Amazon-Book		Last-FM		Yelp2018	
	recall	ndcg	recall	ndcg	recall	ndcg
FM	0.1345	0.0886	0.0778	0.1181	0.0627	0.0768
NFM	**0.1366**	0.0913	**0.0829**	0.1214	0.0660	0.0810
CKE	0.1343	0.0885	0.0736	0.1184	0.0657	0.0805
CFKG	0.1142	0.0770	0.0723	0.1143	0.0522	0.0644
MCRec	0.1113	0.0783	-	-	-	-
RippleNet	0.1336	0.0910	0.0791	0.1238	**0.0664**	**0.0822**
GC-MC	0.1316	0.0874	0.0818	**0.1253**	0.0659	0.0790
KGAT	**0.1489***	**0.1006***	**0.0870***	**0.1325***	**0.0712***	**0.0867***
%Improv.	8.95%	10.05%	4.93%	5.77%	7.18%	5.54%

4.3 Performance Comparison (RQ1)

We first report the performance of all the methods, and then investigate how the modeling of high-order connectivity alleviate the sparsity issues.

4.3.1 ***Overall Comparison.*** The performance comparison results are presented in Table 2. We have the following observations:

- KGAT consistently yields the best performance on all the datasets. In particular, KGAT improves over the strongest baselines *w.r.t.* recall@20 by 8.95%, 4.93%, and 7.18% in Amazon-book, Last-FM, and Yelp2018, respectively. By stacking multiple attentive embedding propagation layers, KGAT is capable of exploring the high-order connectivity in an explicit way, so as to capture collaborative signal effectively. This verifies the significance of capturing collaborative signal to transfer knowledge. Moreover, compared with GC-MC, KGAT justifies the effectiveness of the attention mechanism, specifying the attentive weights *w.r.t.* compositional semantic relations, rather than the fixed weights used in GC-MC.
- SL methods (*i.e.,* FM and NFM) achieve better performance than the CFKG and CKE in most cases, indicating that regularization-based methods might not make full use of item knowledge. In particular, to enrich the representation of an item, FM and NFM exploit the embeddings of its connected entities, while CFKG and CKE only use that of its aligned entities. Furthermore, the cross features in FM and NFM actually serve as the second-order connectivity between users and entities, whereas CFKG and CKE model connectivity on the granularity of triples, leaving high-order connectivity untouched.
- Compared to FM, the performance of RippleNet verifies that incorporating two-hop neighboring items is of importance to enrich user representations. It therefore points to the positive effect of modeling the high-order connectivity or neighbors. However, RippleNet slightly underperforms NFM in Amazon-book and Last-FM, while performing better in Yelp2018. One possible reason is that NFM has stronger expressiveness, since the hidden layer allows NFM to capture the nonlinear and complex feature interactions between user, item, and entity embeddings.
- RippleNet outperforms MCRec by a large margin in Amazon-book. One possible reason is that MCRec depends heavily on the quality of meta-paths, which require extensive domain knowledge to define. The observation is consist with [29].
- GC-MC achieves comparable performance to RippleNet in Last-FM and Yelp2018 datasets. While introducing the high-order connectivity into user and item representations, GC-MC forgoes the semantic relations between nodes; whereas RippleNet utilizes relations to guide the exploration of user preferences.

Table 3: Effect of embedding propagation layer numbers (L).

	Amazon-Book		Last-FM		Yelp2018	
	recall	ndcg	recall	ndcg	recall	ndcg
KGAT-1	0.1393	0.0948	0.0834	0.1286	0.0693	0.0848
KGAT-2	0.1464	0.1002	0.0863	0.1318	0.0714	**0.0872**
KGAT-3	0.1489	0.1006	0.0870	0.1325	0.0712	0.0867
KGAT-4	**0.1503**	**0.1015**	**0.0871**	**0.1329**	**0.0722**	0.0871

4.3.2 ***Performance Comparison w.r.t. Interaction Sparsity Levels.*** One motivation to exploiting KG is to alleviate the sparsity issue, which usually limits the expressiveness of recommender systems. It is hard to establish optimal representations for inactive users with few interactions. Here we investigate whether exploiting connectivity information helps alleviate this issue.

Towards this end, we perform experiments over user groups of different sparsity levels. In particular, we divide the test set into four groups based on interaction number per user, meanwhile try to keep different groups have the same total interactions. Taking Amazon-book dataset as an example, the interaction numbers per user are less than 7, 15, 48, and 4475 respectively. Figure 3 illustrates the results *w.r.t.* ndcg@20 on different user groups in Amazon-book, Last-FM, and Yelp2018. We can see that:

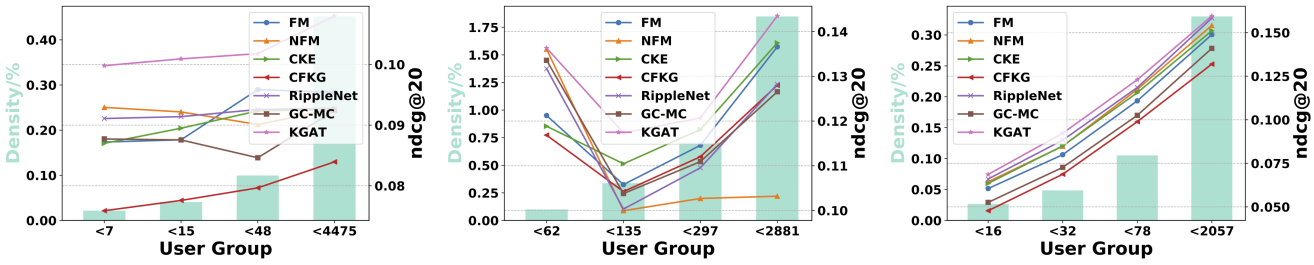
- KGAT outperforms the other models in most cases, especially on the two sparsest user groups in Amazon-Book and Yelp2018. It again verifies the significance of high-order connectivity modeling, which 1) contains the lower-order connectivity used in baselines, and 2) enriches the representations of inactive users via recursive embedding propagation.
- It is worthwhile pointing out that KGAT slightly outperforms some baselines in the densest user group (*e.g.,* the < 2057 group of Yelp2018). One possible reason is that the preferences of users with too many interactions are too general to capture. High-order connectivity could introduce more noise into the user preferences, thus leading to the negative effect.

4.4 Study of KGAT (RQ2)

To get deep insights on the attentive embedding propagation layer of KGAT, we investigate its impact. We first study the influence of layer numbers. In what follows, we explore how different aggregators affect the performance. We then examine the influence of knowledge graph embedding and attention mechanism.

4.4.1 ***Effect of Model Depth.*** We vary the depth of KGAT (*e.g.,* L) to investigate the efficiency of usage of multiple embedding propagation layers. In particular, the layer number is searched in the range of $\{1, 2, 3, 4\}$; we use KGAT-1 to indicate the model with one layer, and similar notations for others. We summarize the results in Table 3, and have the following observations:

- Increasing the depth of KGAT is capable of boosting the performance substantially. Clearly, KGAT-2 and KGAT-3 achieve consistent improvement over KGAT-1 across all the board. We attribute the improvements to the effective modeling of high-order relation between users, items, and entities, which is carried by the second- and third-order connectivities, respectively.



(a) ndcg on Amazon-Book (b) ndcg on Last-FM (c) ndcg on Yelp2018

Figure 3: Performance comparison over the sparsity distribution of user groups on different datasets. The background histograms indicate the density of each user group; meanwhile, the lines demonstrate the performance *w.r.t.* ndcg@20.

Table 4: Effect of aggregators.

Aggregator	Amazon-Book recall	Amazon-Book ndcg	Last-FM recall	Last-FM ndcg	Yelp2018 recall	Yelp2018 ndcg
GCN	0.1381	0.0931	0.0824	0.1278	0.0688	0.0847
GraphSage	0.1372	0.0929	0.0822	0.1268	0.0666	0.0831
Bi-Interaction	**0.1393**	**0.0948**	**0.0834**	**0.1286**	**0.0693**	**0.0848**

Table 5: Effect of knowledge graph embedding and attention mechanism.

	Amazon-Book recall	Amazon-Book ndcg	Last-FM recall	Last-FM ndcg	Yelp2018 recall	Yelp2018 ndcg
w/o K&A	0.1367	0.0928	0.0819	0.1252	0.0654	0.0808
w/o KGE	0.1380	0.0933	0.0826	0.1273	0.0664	0.0824
w/o Att	0.1377	0.0930	0.0826	0.1270	0.0657	0.0815

- Further stacking one more layer over KGAT-3, we observe that KGAT-4 only achieve marginal improvements. It suggests that considering third-order relations among entities could be sufficient to capture the collaborative signal, which is consistent to the findings in [14, 33].
- Jointly analyzing Tables 2 and 3, KGAT-1 consistently outperforms other baselines in most cases. It again verifies the effectiveness of that attentive embedding propagation, empirically showing that it models the first-order relation better.

4.4.2 ***Effect of Aggregators.*** To explore the impact of aggregators, we consider the variants of KGAT-1 that uses different settings — more specifically GCN, GraphSage, and Bi-Interaction (*cf.* Section 3.1), termed KGAT-1_{GCN}, KGAT-$1_{\text{GraphSage}}$, and KGAT-1_{Bi}, respectively. Table 4 summarizes the experimental results. We have the following findings:

- KGAT-1_{GCN} is consistently superior to KGAT-$1_{\text{GraphSage}}$. One possible reason is that GraphSage forgoes the interaction between the entity representation $\mathbf{e}_h$ and its ego-network representation $\mathbf{e}_{\mathcal{N}_h}$. It hence illustrates the importance of feature interaction when performing information aggregation and propagation.
- Compared to KGAT-1_{GCN}, the performance of KGAT-1_{Bi} verifies that incorporating additional feature interaction can improve the representation learning. It again illustrates the rationality and effectiveness of Bi-Interaction aggregator.

4.4.3 ***Effect of Knowledge Graph Embedding and Attention Mechanism.*** To verify the impact of knowledge graph embedding and attention mechanism, we do ablation study by considering three variants of KGAT-1. In particular, we disable the TransR embedding component (*cf.* Equation (2)) of KGAT, termed KGAT-$1_{\text{w/o KGE}}$; we disable the attention mechanism (*cf.* Equation (4)) and set $\pi(h,r,t)$ as $1/|\mathcal{N}_h|$, termed KGAT-$1_{\text{w/o Att}}$. Moreover, we obtain another variant by removing both components, named KGAT-$1_{\text{w/o K\&A}}$. We summarize the experimental results in Table 5 and have the following findings:

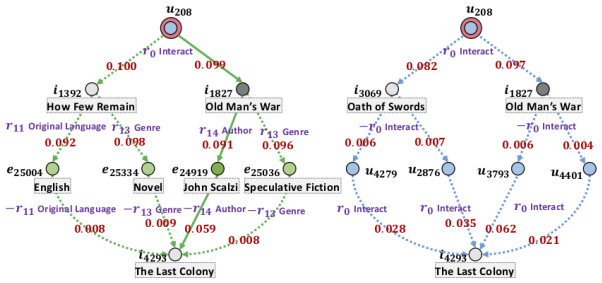


Figure 4: Real Example from Amazon-Book.

- Removing knowledge graph embedding and attention components degrades the model's performance. KGAT-$1_{\text{w/o K\&A}}$ consistently underperforms KGAT-$1_{\text{w/o KGE}}$ and KGAT-$1_{\text{w/o Att}}$. It makes sense since KGAT$_{\text{w/o K\&A}}$ fails to explicitly model the representation relatedness on the granularity of triplets.
- Compared with KGAT-$1_{\text{w/o Att}}$, KGAT-$1_{\text{w/o KGE}}$ performs better in most cases. One possible reason is that treating all neighbors equally (*i.e.,* KGAT-$1_{\text{w/o Att}}$) might introduce noises and mislead the embedding propagation process. It verifies the substantial influence of graph attention mechanism.

4.5 Case Study (RQ3)

Benefiting from the attention mechanism, we can reason on high-order connectivity to infer the user preferences on the target item, offering explanations. Towards this end, we randomly selected one user u_{208} from Amazon-Book, and one relevant item i_{4293} (from the test, unseen in the training phase). We extract behavior-based and attribute-based high-order connectivity connecting the user-item pair, based on the attention scores. Figure 4 shows the visualization of high-order connectivity. There are two key observations:

- KGAT captures the behavior-based and attribute-based high-order connectivity, which play a key role to infer user preferences. The retrieved paths can be viewed as the evidence why the item meets the user's preference. As we can see, the connectivity $u_{208} \xrightarrow{r_0}$ *Old Man's War* $\xrightarrow{r_{14}}$ *John Scalzi* $\xrightarrow{-r_{14}} i_{4293}$ has the highest attention score, labeled with the solid line in the left

subfigure. Hence, we can generate the explanation as *The Last Colony is recommended since you have watched Old Man's War written by the same author John Scalzi.*

- The quality of item knowledge is of crucial importance. As we can see, entity *English* with relation *Original Language* is involved in one path, which is too general to provide high-quality explanations. This inspires us to perform hard attention to filter less informative entities out in future work.

5 CONCLUSION AND FUTURE WORK

In this work, we explore high-order connectivity with semantic relations in CKG for knowledge-aware recommendation. We devised a new framework KGAT, which explicitly models the high-order connectivities in CKG in an end-to-end fashion. At it core is the attentive embedding propagation layer, which adaptively propagates the embeddings from a node's neighbors to update the node's representation. Extensive experiments on three real-world datasets demonstrate the rationality and effectiveness of KGAT.

This work explores the potential of graph neural networks in recommendation, and represents an initial attempt to exploit structural knowledge with information propagation mechanism. Besides knowledge graph, many other structural information indeed exists in real-world scenarios, such as social networks and item contexts. For example, by integrating social network with CKG, we can investigate how social influence affects the recommendation. Another exciting direction is the integration of information propagation and decision process, which opens up research possibilities of explainable recommendation.

Acknowledgement: This research is part of NExT++ research and also supported by the Thousand Youth Talents Program 2018. NExT++ is supported by the National Research Foundation, Prime Minister's Office, Singapore under its IRC@SG Funding Initiative.

REFERENCES

[1] Qingyao Ai, Vahid Azizi, Xu Chen, and Yongfeng Zhang. 2018. Learning Heterogeneous Knowledge Base Embeddings for Explainable Recommendation. *Algorithms* 11, 9 (2018), 137.
[2] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *NeurIPS*. 2787–2795.
[3] Yixin Cao, Lei Hou, Juanzi Li, and Zhiyuan Liu. 2018. Neural Collective Entity Linking. In *COLING*. 675–686.
[4] Yixin Cao, Lei Hou, Juanzi Li, Zhiyuan Liu, Chengjiang Li, Xu Chen, and Tiansi Dong. 2018. Joint Representation Learning of Cross-lingual Words and Entities via Attentive Distant Supervision. In *EMNLP*. 227–237.
[5] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019. Unifying Knowledge Graph Learning and Recommendation: Towards a Better Understanding of User Preferences. In *WWW*.
[6] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 2017. Attentive Collaborative Filtering: Multimedia Recommendation with Item- and Component-Level Attention. In *SIGIR*. 335–344.
[7] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. In *DLRS@RecSys*. 7–10.
[8] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*. 249–256.
[9] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS*. 1025–1035.
[10] Ruining He and Julian McAuley. 2016. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. In *WWW*. 507–517.
[11] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In *SIGIR*. 355–364.
[12] Xiangnan He, Zhankui He, Jingkuan Song, Zhenguang Liu, Yu-Gang Jiang, and Tat-Seng Chua. 2018. NAIS: Neural Attentive Item Similarity Model for Recommendation. *TKDE* 30, 12 (2018), 2354–2366.
[13] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *WWW*. 173–182.
[14] 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*. 1531–1540.
[15] Jin Huang, Wayne Xin Zhao, Hong-Jian Dou, Ji-Rong Wen, and Edward Y. Chang. 2018. Improving Sequential Recommendation with Knowledge-Enhanced Memory Networks. In *SIGIR*. 505–514.
[16] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014).
[17] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.
[18] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems. In *KDD*. 1754–1763.
[19] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *AAAI*. 2181–2187.
[20] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, Vol. 30. 3.
[21] Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. 2018. DeepInf: Social Influence Prediction with Deep Learning. In *KDD*. 2110–2119.
[22] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*. 452–461.
[23] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. Fast context-aware recommendations with factorization machines. In *SIGIR*. 635–644.
[24] Ying Shan, T. Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and J. C. Mao. 2016. Deep Crossing: Web-Scale Modeling without Manually Crafted Combinatorial Features. In *KDD*. 255–262.
[25] 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.
[26] Rianne van den Berg, Thomas N. Kipf, and Max Welling. 2017. Graph Convolutional Matrix Completion. In *KDD*.
[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NeurIPS*. 6000–6010.
[28] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.
[29] 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*. 417–426.
[30] Xiang Wang, Xiangnan He, Fuli Feng, Liqiang Nie, and Tat-Seng Chua. 2018. TEM: Tree-enhanced Embedding Model for Explainable Recommendation. In *WWW*. 1543–1552.
[31] Xiang Wang, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. 2017. Item Silk Road: Recommending Items from Information Domains to Social Users. In *SIGIR*. 185–194.
[32] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *SIGIR*.
[33] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable Reasoning over Knowledge Graphs for Recommendation. In *AAAI*.
[34] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation Learning on Graphs with Jumping Knowledge Networks. In *ICML*, Vol. 80. 5449–5458.
[35] Jheng-Hong Yang, Chih-Ming Chen, Chuan-Ju Wang, and Ming-Feng Tsai. 2018. HOP-rec: high-order proximity for implicit recommendation. In *RecSys*. 140–144.
[36] Xiao Yu, Xiang Ren, Quanquan Gu, Yizhou Sun, and Jiawei Han. 2013. Collaborative filtering with entity similarity regularization in heterogeneous information networks. *IJCAI* 27 (2013).
[37] 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.
[38] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative Knowledge Base Embedding for Recommender Systems. In *KDD*. 353–362.
[39] Huan Zhao, Quanming Yao, Jianda Li, Yangqiu Song, and Dik Lun Lee. 2017. Meta-Graph Based Recommendation Fusion over Heterogeneous Information Networks. In *KDD*. 635–644.
[40] Guorui Zhou, Xiaoqiang Zhu, Chengru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep Interest Network for Click-Through Rate Prediction. In *KDD*. 1059–1068.