VRKG4Rec: Virtual Relational Knowledge Graph for Recommendation

Lingyun Lu
School of Electronic Information and
Communications, Huazhong
University of Science and Technology
Wuhan, China
lulingyun@hust.edu.cn

Bang Wang
School of Electronic Information and
Communications, Huazhong
University of Science and Technology
Wuhan, China
wangbang@hust.edu.cn

Zizhuo Zhang
School of Electronic Information and
Communications, Huazhong
University of Science and Technology
Wuhan, China
zhangzizhuo@hust.edu.cn

Shenghao Liu
School of Cyber Science and
Engineering, Huazhong University of
Science and Technology
Wuhan, China
liushenghao@hust.edu.cn

Han Xu
School of Journalism and Information
Communication, Huazhong
University of Science and Technology
Wuhan, China
xuh@hust.edu.cn

ABSTRACT

Incorporating knowledge graph as side information has become a new trend in recommendation systems. Recent studies regard items as entities of a knowledge graph and leverage graph neural networks to assist item encoding, yet by considering each relation type independently. However, relation types are often too many and sometimes one relation type involves too few entities. We argue that there may exist some latent relevance among relations in KG. It may not necessary nor effective to consider all relation types for item encoding. In this paper, we propose a VRKG4Rec model (<u>V</u>irtual <u>R</u>elational <u>K</u>nowledge <u>G</u>raphs for <u>R</u>ecommendation), which clusters relations with latent relevance to generates virtual relations. Specifically, we first construct virtual relational graphs (VRKGs) by an unsupervised learning scheme. We also design a *local weighted smoothing* (LWS) mechanism for node encoding on VRKGs, which iteratively updates a node embedding only depending on the node itself and its neighbors, but involve no additional training parameters. LWS mechanism is also employed on a user-item bipartite graph for user representation learning, which utilizes item encodings with virtual relational knowledge to help train user representations. Experiment results on two public datasets validate that our VRKG4Rec model outperforms the state-of-the-art methods. The implementations are available at https://github.com/lulu0913/VRKG4Rec.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Mathematics of computing** → *Graph algorithms*.

KEYWORDS

recommendation system; knowledge graph; graph neural network

1 INTRODUCTION

Recommender system (RS) [7, 10, 33, 38] as an information filtering tool plays an important role in our daily life to deal with the problem of information overload [6]. Traditional RS mainly depends on collaborative filtering of user-item interaction records, which suffers from the data sparsity problem [17, 18, 34, 39]. Recently, knowledge graph (KG) [9, 28] with entities and rich relation connections has been exploited in RS to serve as side information for alleviating this problem [43, 44]. The entities in KG can describe the attributes of the item, thus enhancing the item content representation. Generally speaking, a KG consists of many knowledge triplets (h, r, t), where h and t are a head entity and a tail entity respectively, and r is the relation from h to t. For example, (*Leonardo*, *star*, *Titanic*) states the fact that Leonardo is the star of the movie Titanic. A KG can contain multiple types of relations, and more than one type of relation can exist between h and t in a KG. This makes it possible for KG to transfer the complex relationships in real world.

The main challenge of KG-based recommendation is how to extract useful information from structured knowledge triplets for better learning item representation. Some researches [1, 26, 35] first learn entity representation in a KG by some knowledge representation learning method (e.g. the TransX series [2, 13, 16, 31]) and use the learned embeddings as pre-trained features in a follow-up recommendation task. Although these methods can well model knowledge triplet relationships in a KG, they mainly focus on the first-order connectivity but ignore the global relationships of entities. Some researches [5, 19] explore a kind of meta-path connecting

history entity and the candidate entity. However, these methods suffer from two main problems: (1) Devising such meta-path requires strong expertise, which has poor transfer capability. (2) It is labor-intensive and time-consuming when applying on the KGs of large scale.

Recently, some researches [24, 25, 29, 32] use graph neural networks (GNNs) for encoding knowledge in terms of entities and their relations into item representation and have achieved excellent and promising performance in recommendation. The key idea of these GNN-based models is to aggregate the information from direct-neighbor into node representation and incorporate the information from high-order neighbors by stacking multiple GNN layers. [36, 40, 41] Despite their significant performance improvements, we argue that they still suffer from a common problem.

Existing methods [23, 25, 27, 29, 30, 32] apply the devised aggregation mechanism directly on the input KG which means KG quality is crucial for a distinguished performance. However, in real-world dataset, the relations in KG always exhibit long-tail distribution. The distribution of KG relations collected from real-world dataset Last-FM[1] is reported in Fig. 1(a), where x-axis is the relation exposure count in KG and y-axis represents the corresponding number of relations. From Fig. 1(a), we notice that the relations in KG show long-tail issue, which means most of the relations appear only a few times. This will affect user preference propagation in KG if we treat these relations as totally irrelevant types of edges. For example, in Fig. 1(b), there are four triplets in KG. They are *triplet 1=(James Cameron, screenwriter, Movie B)*, *triplet 2=(James Cameron, direct, Titanic)*, *triplet 3=(Leonardo, star, Titanic)* and *triplet 4=(Leonardo, star, Movie A)*. It can be deduced from *triplet 3* and *triplet 4* that *Movie A* will be recommended to Mary since Mary has watched *Titanic* and it has the same movie star *Leonardo*, while *Movie B* may not recommended to Mary for *triplet 1* and *triplet 2* have different relations, that is *direct* and *screenwriter*. However, *direct* and *screenwriter* have the similar semantic. So Mary will also interested in *Movie B*. As a result, it is not wise to consider every relation independently, especially in long-tail situation that most relations only exist in a few triplets.

To deal with the problem, in this paper, we propose to generate virtual relations for KG and design a Virtual Relational Knowledge Graph for Recommendation model (VRKG4Rec). We first construct *virtual relational knowledge graphs* (VRKGs) from an unsupervised learning approach. Given entities contribute differently to item, we next devise a local weighted smoothing (LWS) mechanism on each VRKG to learn aggregated relational knowledge for item encoding while keeping the semantic independence. We obtain the final representation by fusing item encodings from all VRKGs. We also employ the LWS mechanism on a user-item bipartite graph to learn user representation, which utilizes item encodings with relational knowledge to help train user representations. Experiment results on two public datasets validate that our VRKG4Rec model outperforms the state-of-the-art methods.

we summarize the contribution of this work as follows:

- Propose to construct Virtual Relational Knowledge Graph (VRKG) through unsupervised learning to reveal the relevant
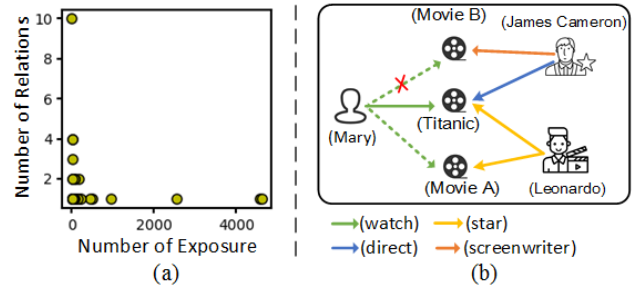
Figure 1: Illustration of two kinds of motivations. (a)The long-tail relation distribution of Last.FM dataset. (b)An illustration example of necessity of considering the relevance of different relations

relations in item encoding and alleviate long-tail issues for recommendation.
- Devise a new aggregation scheme called Local Weighted Smoothing (LWS) mechanism to aggregate weighted neighbor knowledge into item representation.
- Conduct empirical studies on two benchmark datasets to demonstrate the proposed algorithm outperforms the state-of-the-art methods.

The rest of the paper is structured as follows: Section 2 gives the definition of the problem in this paper. The proposed VRKG4Rec model is presented in Section 3 and experimented in Section 4. We briefly reviews the related work in Section 5. Section 6 concludes the paper with some discussions.

2 PROBLEM FORMULATION

Knowledge-aware recommendation task is formulated as follows. The inputs of the recommendation model are interaction data in between users and items and a knowledge graph; while the output is a similarity score of a target user and a candidate item.

The **interaction data** indicates users' interactions with items. There are total M number of users and N number of items. Let $\mathcal{U} = u_1, u_2, \cdots, u_M$ and $\mathcal{I} = i_1, i_2, \cdots, i_N$ be a set of users and a set of items, respectively. Let $O^+ = \{(u, i) \mid u \in \mathcal{U}, i \in \mathcal{I}\}$ denote a set of positive interactions. Following the previous study KGAT [29], we view the interaction as relation r^+ and we can construct user-item bipartite graph $\mathcal{G}^I = \{(u, r^+, i)|(u, i) \in O^+\}$. Note that an item i in $\mathcal{G}^I$ can be aligned with an entity h in knowledge graph.

The **knowledge graph** consists of many triplets which states real-world facts, such as the attributes of items or the relationship between entities. We use $\mathcal{G} = \{(h, r, t) \mid h, t \in \mathcal{E}, r \in \mathcal{R}\}$ to denote knowledge graph, where $\mathcal{E}$ is the set of entities and $\mathcal{R}$ is the relation set, which contains both canonical directions *(e.g. Star)* and the inverse directions *(e.g. StaredBy)* ; h, r and t is the head entity, relation and tail entity of the knowledge triplet respectively. In recommendation task, we align the items with entities ($\mathcal{I} \subset \mathcal{E}$) in knowledge graph so as to utilize the side information to facilitate item representation.
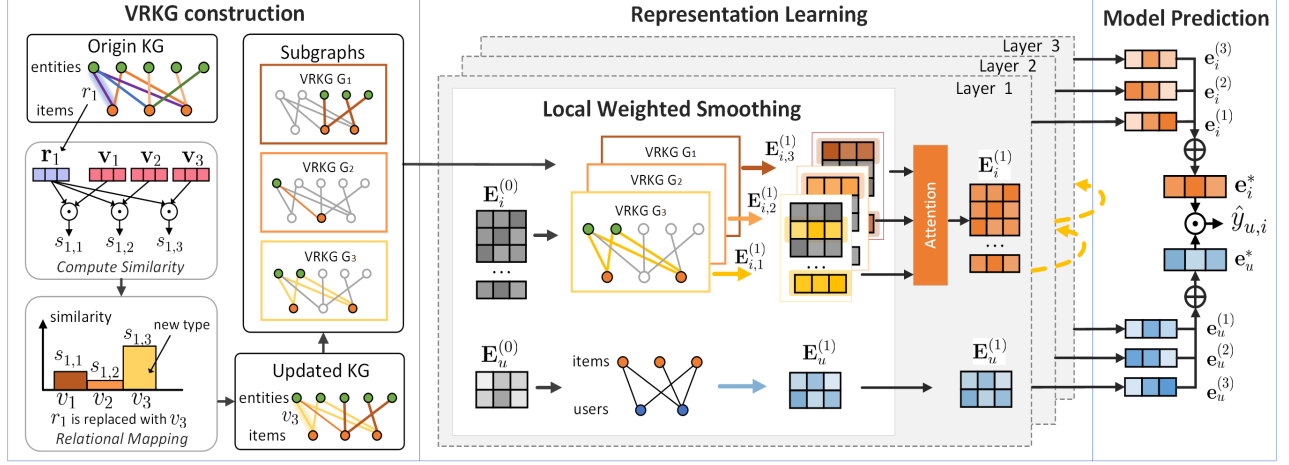
Figure 2: Overview of the proposed VRKG4Rec model

3 THE PROPOSED MODEL

In this section, we illustrate our VRKG4Rec model in detail and present the overview of the model in Fig. 2. The proposed model consists of three key components: (1) Virtual Relational Knowledge Graph (VRKG) construction, which clusters all the relations in origin KG into K virtual relations by unsupervised learning and construct K VRKGs. (2) Local Weighted Smoothing (LWS) aggregation module, which learns the representation of a node by weighted smoothing the representation of neighbor nodes to makes its representation closer to the most similar neighbors. (3) Representation learning module, which applies LWS to generate use and item representation.

3.1 VRKG Construction

In this section, we first reveal the relevance relations for item encoding by clustering all types of original relations into K virtual relations. Then we construct virtual relational knowledge graphs based on these virtual relations to demonstrate item in terms of different attributes.

For the purpose of clustering relevant relations, we propose an unsupervised learning method to explore the latent factor of each original relation and fuse the original relations with similar latent factor into one kind of virtual relation. Firstly, we initialize the representations of such virtual relations as a virtual centroid matrix $V \in \mathbb{R}^{K \times d}$

$$V = (v_1, v_2, ..., v_K)^\top, \tag{1}$$

where the k-th row $v_k \in \mathbb{R}^d$ is the representation of the k-th virtual relation. K is the number of virtual relations and is a hyper-parameter which is set as 3 by default.

On basis of this, we compute the similarities between the original relations in KG with the virtual relations. For the relation $r_p \in \mathcal{R}$, we construct its similarity vector $s_p \in \mathbb{R}^K$ by

$$s_p = (g(r_p, v_1), g(r_p, v_2), ..., g(r_p, v_K)) \tag{2}$$

$$g(r_p, v_k) = r_p^\top v_k \tag{3}$$

where $r_p \in \mathbb{R}^d$ is the embedding of relation r_p, and $g(\cdot)$ is a similarity function, here we use inner product function.

The relation r_p is replaced with a virtual relation $v_{k'}$ with the highest similarity; While the knowledge graph $\mathcal{G}$ will be updated as $\mathcal{G}'$ accordingly.

$$k' = \arg \max s_p \tag{4}$$

$$= \arg \max_{k=1,2,...,K} (g(r_p, v_1), ..., g(r_p, v_k), ...) \tag{5}$$

$$(h, r_p, t) \leftarrow (h, v_{k'}, t). \tag{6}$$

The virtual centroid matrix V is trained together with the training of our recommendation model parameters Θ. We note that such a joint training of V and Θ can help construct virtual relations more suitable for a downstream recommendation task.

After model training, we divide the origin KG $\mathcal{G}$ into some subgraphs, called virtual relational knowledge graphs (VRKGs), to capture the connection information between relations and preserve the semantic independence of item attributes. Specifically, VRKGs are constructed based on the updated knowledge graph $\mathcal{G}'$, each of which contains one of virtual relation and its connected entities from the knowledge graph $\mathcal{G}'$.

$$\mathcal{G}_k = \{(h, r', t) \mid (h, r', t) \in \mathcal{G}', r' = v_k\}. \tag{7}$$

3.2 Local Weighted Smoothing

For obtained VRKGs, We design a *Local Weighted Smoothing* (LWS) mechanism to encode local information from direct neighbors into node embedding. The basic idea is to iteratively update the vector of a node closer to its direct neighbors that are similar in the embedding space and optimize the item embeddings to make them most advantageous for recommendation. The upper right part of Fig. 3 shows the core operation of the LWS mechanism with single iteration.

For an entity h in $\mathcal{G}_k$, the neighbor node set of h is denoted as $\mathcal{N}_k(h) = \{t|(h, v_k, t) \in \mathcal{G}_k\}$. To characterize the topology structure of VRKG $\mathcal{G}_k$, we proposed to weighted smooth the embedding of
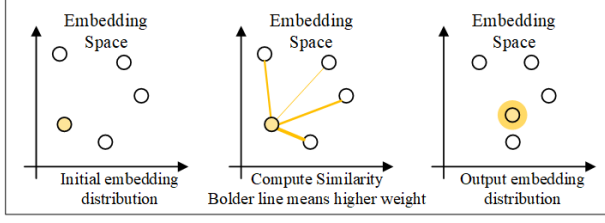
Figure 3: Core operation of LWS with single iteration

neighbor nodes over the propagation:

$$e_{\mathcal{N}_k(h)}^{(0)} = \sum_{t \in \mathcal{N}_k(h)} \pi(h,t) e_t^{(0)}, \tag{8}$$

where $e_t^{(0)}$ is the ID embedding of entity t, $\pi(h,t)$ is the weight of t in local information propagation. The weight reflects the contribution of entity t to depicting the attributes of entity h. We argue that the more important entity t is, the more information should be propagated and entity h need to be closer to t in embedding space. For this purpose, we compute the similarity between entity t and h to serve as the weight $\pi(h,t)$. Here, we use inner product to measure the similarity for simplicity:

$$\pi(h,t) = e_h^{(0)\top} \cdot e_t^{(0)}, \tag{9}$$

where $e_h^{(0)}$ is the ID embedding of entity h.

Then we aggregate the entity representation e_h and the local information representation $e_{\mathcal{N}_k(h)}$ to obtain a temp representation as $u_h^{(1)}$:

$$u_h^{(1)} = \text{AGG}(e_h^{(0)}, e_{\mathcal{N}_k(h)}^{(0)}) \tag{10}$$

$$= \text{NORM}(e_h^{(0)} + e_{\mathcal{N}_k(h)}^{(0)}), \tag{11}$$

We simply sum two representations up to update the entity representation without introduce additional training parameters in aggregator. Note that we didn't normalize the weight by adopting the softmax function in aggregation step like most knowledge-aware graph neural networks (e.g. KGAT [29], KGCN [27]) do. In this way, we can simplify the calculation. More importantly, it frees us from sampling a fixed number of neighbors for each node due to the limit of softmax computing and this enables us to take full advantage of knowledge graph without information loss in sampling. However, skipping normalization step may lead to the scale of the updated vector getting larger and becoming incomparable. To solve this problem, we normalize the temp vector $u_h^{(1)}$ by:

$$\text{NORM}(u) = \frac{u}{\|u\|} \cdot \frac{\|u\|^2}{\|u\|^2 + 1}, \tag{12}$$

where $\|u\|$ is 2-norm of vector u. In this way, we can norm the scale of the vector to the range from 0 to 1, keeping its direction and relative sizes of vectors. We use $f_{agg}(\cdot)$ to denote the aggregation function.

$$u_{h,k}^{(1)} = f_{agg}\left(\{(e_h^{(0)}, e_t^{(0)}) \mid t \in \mathcal{N}_k(h)\}\right) \tag{13}$$

To further modify the representation of the node and make the distance in embedding space better reflects the similarity, we iteratively smooth the node vector based on the temp representation $u_h^{(1)}$ by repeat the above weighed propagation step:

$$u_{h,k}^{(q)} = f_{agg}\left(\{(u_{h,k}^{(q-1)}, e_t^{(0)}) \mid t \in \mathcal{N}_k(h)\}\right), \tag{14}$$

where $q = 2, 3, ..., Q$. We set $Q = 3$ as default in our experiments.

Up to now, we finally update the representation of entity h as $e_{h,k}^{(1)} = u_{h,k}^{(Q)}$. Note that during each iteration, we only smooth the vector of h, keeping the neighbor vector unchanged. We use $f_{LWS}(\cdot)$ to denote the entire smooth operation of incorporing the first-order information into node embedding as follows:

$$e_{h,k}^{(1)} = f_{LWS}\left(\{(e_h^{(0)}, e_t^{(0)}) \mid t \in \mathcal{N}_k(h)\}\right), \tag{15}$$

where $e_{h,k}^{(1)}$ is the updated representation of entity h in VKRG $\mathcal{G}_k$. So far, we are able to encode the first-order information supplied by $\mathcal{G}_k$ into item representation, forcing it closer to similar entity in embedding space.

3.3 Representation Learning

3.3.1 Item Representation Learning. To preserve the semantic independence of the item attributes, we apply LWS on each of the K VRKGs respectively to encode item from K different aspects and then fuse these K embeddings by attention mechanism to acquire final item representation as $e_h^{(1)}$:

$$e_h^{(1)} = \alpha_1 e_{h,1}^{(1)} + \alpha_2 e_{h,2}^{(1)} + ... + \alpha_K e_{h,K}^{(1)} \tag{16}$$

$$e_h^{(1)} = \sum_{k=1}^{K} \alpha_k f_{LWS}\left(\{(e_h^{(0)}, e_t^{(0)}) \mid t \in \mathcal{N}_k(h)\}\right), \tag{17}$$

where $e_h^{(1)} \in \mathbb{R}^d$ is the representation that collects the first-order connectivity information from knowledge graph. α_k is the weight of the representation learned from VRKG $\mathcal{G}_k$. It denotes the importance of this virtual relation in item encoding and is learned during training process.

In order to incorporate the information from high-order neighbors, we further stack more LWS aggregation layers. More formally, we recursively formulate the representation of entity h after l layers as:

$$e_h^{(l)} = \sum_{k=1}^{K} \alpha_k f_{LWS}\left(\{(e_h^{(l-1)}, e_t^{(l-1)}) \mid t \in \mathcal{N}_k(h)\}\right), \tag{18}$$

where $e_h^{(l)}$ denotes the representation of entity h, which contains the information from its l-hop neighbors, and $l = 1, 2, ..., L$. We set $L = 3$ as default in our experiment.

3.3.2 User Representation Learning. To obtain user representation, we first construct a user-item bipartite graph based on the interaction data, which is denoted as $\mathcal{G}^I = \{(u, r^+, i) \mid (u, i) \in O^+\}$. Specifically, there is an edge in between u and i, only when the pair $(u, i) \in O^+$.

Then, in order to make use of collaborative information and discriminate the impact of items that user has interacted before, we apply LWS on $\mathcal{G}^I$ to unequally encode the information from historical interactions into user representation:

$$e_u^{(l)} = f_{LWS}\left(\{(e_u^{(l-1)}, e_i^{(l-1)}) \mid i \in \mathcal{N}(u)\}\right), \tag{19}$$

Algorithm 1: *Local Weighted Smoothing* (LWS) Algorithm for user and item representation (taking user u and item h for example)

Input: VRKG $\mathcal{G}_k$ $(k = 1, ..., K)$; user-item interation graph $\mathcal{G}^I$; trainable parameters: $e_h, e_u, \{e_t | t \in \mathcal{N}_k(h)\}_{k=1}^K$ and $\{e_i | i \in \mathcal{N}(u)\}$, α_k

Output: e_u^* and e_h^*

1   $e_h^{(0)} \leftarrow e_h$; $e_u^{(0)} \leftarrow e_u$; $e_t^{(0)} \leftarrow e_t$; $e_i^{(0)} \leftarrow e_i$

2   **for** $l = 1,2,...,L$ **do**

3   /*Item Representation Learning*/

4   **for** $k = 1,2,...,K$ **do**

5         $u_{h,k}^{(0)} \leftarrow e_h^{(l-1)}$

6   **for** $q = 1,2,...,Q$ **do**

7            $\pi(h,t) \leftarrow u_{h,k}^{(q-1)\top} \cdot e_t^{(l-1)}$

8            $u_{h,k}^{(q)} \leftarrow$
              $\text{NORM}(u_{h,k}^{(q-1)} + \sum_{t \in \mathcal{N}_k(h)} \pi(h,t) e_t^{(l-1)})$

9   **end**

10         $e_{h,k}^{(l)} \leftarrow u_{h,k}^{(T)}$

11   **end**

12      $e_h^l \leftarrow \sum_{k=1}^K e_{h,k}^{(l)}$

13

14   /*User Representation Learning*/

15      $u_u^{(0)} \leftarrow e_u^{(l-1)}$

16   **for** $q = 1,2,...,Q$ **do**

17         $\pi(u,i) \leftarrow u_u^{(q-1)\top} \cdot e_i^{(l-1)}$

18         $u_u^{(q)} \leftarrow \text{NORM}(u_u^{(q-1)} + \sum_{t \in \mathcal{N}(i)} \pi(u,i) e_i^{(l-1)})$

19   **end**

20      $e_u^{(l)} \leftarrow u_u^{(T)}$

21   **end**

Result: $e_u^* \leftarrow \sum_{l=0}^L \alpha_k e_u^{(l)}$; $e_i^* \leftarrow \sum_{l=0}^L e_i^{(l)}$

---

where $l = 1, 2, ..., L$, $\mathcal{N}(u)$ is a set of items that a user u has interacted with, $e_u^{(0)}$ and $e_i^{(0)}$ denote the ID embedding of the user u and item i respectively. Note that item i is aligned with entity h in KG and they have the same embedding, that is $e_i^{(l)} = e_h^{(l)}$, where $l = 1, 2, ..., L$.

It worth mentioning that item nodes that both appear in the knowledge graph $\mathcal{G}$ and the bipartite graph $\mathcal{G}^I$ serve as a bridge to combine the two graphs. This allows information to be passed between graphs seamlessly. We propagate information on both graphs in parallel and train user and item representation simultaneously. By stacking multiple LWS layers, high-order relational knowledge from items can be propagated to user representation towards better recommendation performance. Algorithm 1 describes the detailed LWS operations for learning user and item representation. We take user u and entity h (aligned with item i) as an example.

3.4 Prediction

After L LWS layers, we obtain the representations of user u and item i (corresponding to node h in knowledge graph) at different

layers. Given each output embedding focuses on the neighbors of different hops, we sum up the output of each layer as the final representation.

$$e_u^* = e_u^{(0)} + ... + e_u^{(L)} \tag{20}$$

$$e_i^* = e_i^{(0)} + ... + e_i^{(L)} \tag{21}$$

Finally, we employ the inner product similarity for prediction as follows:

$$\hat{y}_{ui} = e_u^{(*)\top} e_i^{(*)}. \tag{22}$$

3.5 Optimization

To train this model, we construct the BPR loss function

$$\mathcal{L}_{BPR} = \sum_{(u,i,j) \in O} -\ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}), \tag{23}$$

where $O = \{(u,i,j) \mid (u,i) \in O^+, (u,j) \in O^-\}$ is the training set. O^+ is the set of positive interaction pairs which consists of historical user-item interactions; While O^- is the set of negative pairs (u,j), where j is randomly sampled from items that user u has never interacted with. $\sigma(\cdot)$ is the sigmoid function.

The objective function for training model parameters is:

$$\mathcal{L} = \mathcal{L}_{BPR} + \lambda \|\Theta\|_2^2, \tag{24}$$

where $\Theta = \{e_u^{(0)}, e_i^{(0)}, r_p, V, R \mid u \in \mathcal{U}, i \in \mathcal{E}, p \in \mathcal{R}\}$ is the model parameter set. $\mathcal{E}$ is the entity set containing item set $\mathcal{I}$, and generally $\mathcal{I} \subseteq \mathcal{E}$. λ is a hyperparameter controlling L_2 regularization to prevent overfitting.

4 EXPERIMENTS

Table 1: Dataset Statistics

dataset	User-Item Interaction			Knowledge Graph		
	#user	#item	#rating	#entity	#relation	#triplet
Last.FM	1,872	3,846	42,346	9,366	60	15,518
Movie-1M	6,036	2,347	753,772	6,729	7	20,195

In this section, we conduct empirical studies to demonstrate the effectiveness of proposed VRKG4Rec. The results of the experiments answer the following research questions:

- **RQ1:** How does VRKG4Rec perform, comparing with the state-of-the-art KG-based approaches?
- **RQ2:** How does the key component, constructing VRKGs, contribute to model performance?
- **RQ3:** How does hyper-parameters (e.g.the number of iterations and the depth of GNN layer) affect model performance?
- **RQ4:** How does VRKG4Rec explore user preference and give an intuitive explainability?

4.1 Experiment settings

4.1.1 Datasets. We conduct our experiments on two public datasets: Last.FM[2] and MovieLens-1M[3], which is widely adopted in recommendation problems. Last.FM is built from Last.fm online music system. It contains music listening records of two thousand users.

---

[2]https://grouplens.org/datasets/hetrec-2011/
[3]https://grouplens.org/datasets/movielens/1m/

Table 2: Overall comparison of performance

Dateset	Model	metric@1 (%)			metric@5 (%)			metric@10 (%)			metric@20 (%)		
		recall	NDCG	HR	recall	NDCG	HR	recall	NDCG	HR	recall	NDCG	HR
Last	FM	1.93	4.40	4.40	5.33	4.67	12.80	8.83	6.07	19.40	14.02	7.72	28.01
	NFM	1.50	3.90	3.90	5.95	4.80	13.20	9.52	6.26	21.10	14.97	8.05	29.90
	CKE	4.43	10.31	10.31	13.06	11.26	26.58	18.85	13.62	35.02	26.95	16.25	46.11
	KGAT	2.42	5.67	5.67	7.86	9.49	16.76	12.56	12.58	25.92	20.59	16.71	37.67
	KGIN	6.06	13.98	13.98	17.42	15.24	35.92	24.96	18.32	47.07	35.49	21.69	59.07
	proposed	**6.79**	**16.34**	**16.34**	**20.15**	**17.62**	**39.84**	**28.05**	**20.85**	**50.69**	**38.78**	**23.02**	**61.84**
	Improv.	+12.05%	+14.44%	+14.44%	+15.67%	+15.62%	+10.91%	+12.38%	+13.81%	+7.69%	+9.27%	+6.13%	+4.69%
ML	FM	3.53	32.70	32.70	11.65	27.57	64.30	19.41	26.88	76.50	29.11	27.98	85.10
	NFM	2.98	27.70	27.70	11.62	24.88	62.40	17.88	23.80	74.40	27.59	24.84	84.30
	CKE	3.85	33.54	33.54	13.62	28.78	66.65	21.19	27.89	78.29	31.30	29.18	86.51
	KGAT	2.63	23.15	23.15	10.03	20.68	57.01	16.98	21.09	71.59	26.37	23.05	82.15
	KGIN	**4.69**	11.99	11.99	**15.14**	12.92	31.22	22.66	15.95	43.22	31.50	19.35	53.22
	proposed	4.29	**36.74**	**36.74**	15.01	**31.38**	**70.13**	**23.29**	**30.53**	**80.55**	**34.12**	**31.92**	**88.34**
	Improv.	-8.69%	+9.54%	+9.54%	-0.85%	+9.03%	+5.22%	+2.78%	+9.47%	+2.89%	+8.31%	+9.39%	+2.12%

MovieLens-1M is a movie recommendation dataset, which consists of approximately 1 million rating data on the Movie-Lens website.

Following the prior studies [25, 26], we convert the record data into user-item pair to serve as positive interaction data and we use Microsoft Satori to construct the knowledge graph for each dataset by matching the item IDs with the head of all triples. The basic statistics of the two datasets are presented in Table 1.

For each dataset, we randomly select 80% of positive interaction data for training, and treat the remaining as the test set.

4.1.2 Competitors. We compare our proposed VRKG4Rec with the following algorithms that also exploit knowledge graph for recommendation.

- **FM** [21] is a factorization model, which considers second-order feature interactions between items and a KG.
- **NFM** [11] is a state-of-the-art factorization model, which modifying FM by using a neural network.
- **CKE** [35] is an embedding-based method, which incorporates knowledge into the MF framework.
- **KGAT** [29] designs the knowledge graph attention network to model high-order conductivities in a KG.
- **KGIN** [30] applies a GNN to explore user-item interactions by using auxiliary item knowledge.

4.1.3 Evaluation metrics. Following previous study KGIN [30], we conduct all-ranking strategy [15] in evaluation. Specifically, For each user in test set, We regard items that the user has interacted as positive items and treat the rest as testing items. We predict the ratings of all these testing items and select the items with highest ratings to form the recommendation list. To evaluate *Top-n* recommendation performance. We adopt widely-used evaluation protocols: Recall, NDCG, Precision and HR. Without specification, we set the length of recommendation list as 20. The result is reported as the average metrics for all users in test set.

4.1.4 Parameter setting. We implement our proposed model in PyTorch. We fix the size of ID embeddings d as 64, the optimizer as Adam [14], and the batch size as 1024. Without specification, we set the number of VRKGs K as 3. Meanwhile, we fix the number of iteration scales Q as 3 and GNN layers L as 2. Besides, the learning

rate is set as $l_r = 10^{-4}$ and the L_2 penalty is set as 10^{-5}. We make a test per 10 epochs and the total epoch number is set as 1000.

4.2 Performance Comparison(RQ1)

We vary the length of the recommendation list in $\{1, 5, 10, 20\}$ to conduct experiments. Table 2 presents the overall performance comparison between our VRKG4Rec and the competitors, where the best results in every column are boldfaced and the second are underlined. It can be observed that our VRKG4Rec outperforms the competitors on both two datasets in most of cases. We conduct some analysis and discussions as follows:

We attribute the better performance of VRKG4Rec to our conversion of many relations of a knowledge graph into a few virtual relations. Such virtual relations can not only find out the relevance relations in KG, but also help encoding those relational knowledge more beneficial to the downstream recommendation task. Besides, our LWS for encoding an item merely from its neighbor embeddings. Such an encoding mechanism focuses on converting local relational knowledge into neighboring item encodings, as it tries to ensure relational knowledge-connected entities with closer distances in an embedding space.

The FM and NFM perform poorly on both datasets, because factorization model is unable to fully exploit the relations of KG to facilitate item representation. The embedding-based method CKE utilizes TransR to encode the first-order entity and relation information of a KG for entity representation learning; While the KGAT and KGIN design graph neural models so as to capture high-order neighbors' information and relational knowledge. This operation, however, can be either constructive or destructive to recommendation.

It can be observed that the KGAT performs rather worse on both two dataset. We note that the KG for Last-FM dataset contains 60 relation types and about 155K knowledge triplets; While the Movielens-1M contains only 15 relation types and about 200K knowledge triplets. Furthermore, both items and entities are fewer in the Movielens-1M dataset. The worse performance of KGAT on the Last-FM dataset suggest that in a sparse KG but with many relation type, it is not wise to consider all relations in item encoding.
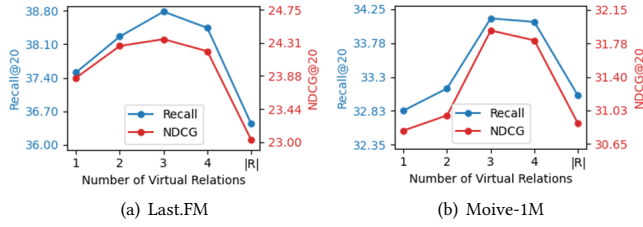
(a) Last.FM

(b) Moive-1M

Figure 4: Impact of VRKGs

And on relatively dense dataset, it may lead to overfit to discriminate the attention of all relations. We note that our VRKG4Rec is robust to different KG scales for its conversion of virtual relations.

KGIN and CKE have opposite performance on the two dataset, where KGIN achieves better performance than CKE on Last.FM and worse than CKE in terms of most metrics on MovieLens-1M. It is because that the KG for MovieLens-1M mainly consists of one-hop knowledge triples and TransR in CKE can better learn this connectivity while KGIN may introduce noise information during high-order propagation.

4.3 Impact of VRKG(RQ2)

In order to evaluate the effectiveness of VRKG construction and virtual relation clustering for item encoding, we consider two KG variants: (1)Variant-1 doesn't construct VRKG but directly apply LWS on the KG without discriminating relation types; (2)Variant-2 discards virtual relation clustering but construct VRKGs based on origin KG. For each relation type, we construct one relational knowledge graph, on which the LWS is applied. To further analyze the impact of virtual relation number, we set the value of K in range of $\{2, 3, 4\}$. We report the performance of *Recall@20* and *NDCG@20* in Fig. 4, where variant-1 is corresponding to $K = 1$ while variant-2 is to $K = |\mathcal{R}|$. Note that $|\mathcal{R}|$ is the number of relations in origin KG. From Fig. 4, we can observe that:

On the one hand, the worse performance of variant-1 indicates the effectiveness of construing VRKGs. We attribute its superiority to two reasons. (1) Variant-1 treats knowledge graph as a homogeneous graph, ignoring the rich relational information. (2) Furthermore, Variant-1 aggregates the information from all neighbors into one embedding, which distorts the semantic independence and integrity of item attributes. On the other hand, the performance of variant-2 goes down dramatically, indicating that not every relation type for item encoding is constructive to recommendation.

We notice that discarding virtual relation clustering has greater impact on Last-FM than Movie-1M dataset. The possible reason is that the KG in Movie-1M doesn't have that many relation types like that in Last-FM and each type of relations is relatively independent for item representation. This again validates the necessity of constructing virtual relations especially on the KG with a large number of edges.

On both two datasets, the curves of Recall and NDCG first increase and then decline after reaching peak performance when $K = 3$. It indicates that clustering the type of relations will help to capture the relevance of relations. However, with the number of virtual relations increasing, some highly relevant entities will
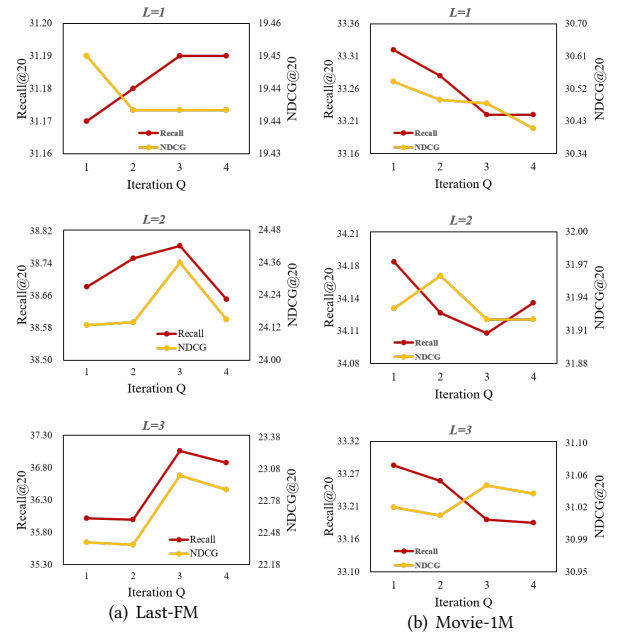

(a) Last-FM

(b) Movie-1M

Figure 5: Impact of iteration Q and Layer L

be separated into different VRKGs in item encoding, which may prevent useful information passing.

4.4 Parameter Sensitivity(RQ3)

In this section, we analyze the impact of the number of LWS iterations Q and the number of the GNN layers L on the performance of VRKG4Rec model. We perform the experiment on Last-FM and MovieLens-1M dataset while setting the number of virtual relations K as 3. Fig. 5 illustrates the performance of *Recall@20* and *NDCG@20* on both two datasets.

In Fig. 5(a) and Fig. 5(b), we fix the number of GNN layer L and change the iteration number Q in range of $\{1, 2, 3, 4\}$. We notice that the performance curve will also first increase and then decline on Last-FM dataset. The result indicates that with Q increase, the item will closer to the similar neighbor in embedding space, which will benefit item representation in recommendation task. Nevertheless, with Q getting bigger, the embeddings of nodes will be too close to discriminate, thus doing harm to model performance. Furthermore, the performance shows a downward trend when iteration Q increase on MovieLens-1M, for the reason that the KG in MovieLens-1M has richer triples and less entities than that in Last-FM and the denser connection make it easier for embedding to be over-smoothing.

If we fix the number of iteration Q and increase L in range of $\{1, 2, 3\}$, the performance will first goes up and then falls down. Specifically, model with $L = 2$ outperforms that with $L = 1$, as the latter only explores the first-order connectivity in KG, leaving the high-order semantic information untouched. However, the performance get worse when $L = 3$, this could be that irrelevant or noise information is generated during long-range propagation.
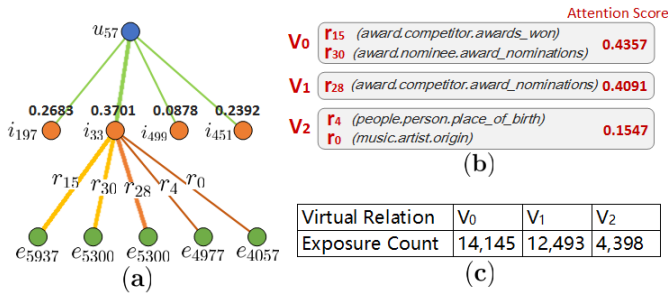
Figure 6: Case study for from Last-FM dataset. (a) shows the preference of user u_{57} on the interacted items and the clustered relations. Best viewed in color. (b) shows semantics of the relations in case study and the corresponding virtual relations. (c) shows exposure count of virtual relations.

Summarily, with the layer L increasing, the model performance will first improve significantly, and then degrade across-the-board regardless of the iteration number Q. We can draw the conclusion that the GNN layser L plays a decisive part in model performance. Because the number of layer L determines the depth of receptive field. Stacking L layers will propagate information from L-hop neighbors into item embedding. While iteration Q fine-tunes the information from 1-hop neighbors to control the smoothness of the item embedding and its neighbor embeddings.

4.5 Case study(RQ4)

In case study, We take user u_{57} and item i_{33} from Last-FM dataset as an example to give an intuitive explainability. We further exhibit the exposure count of virtual relations to validate effectiveness of alleviating the long-tail issue. Fig. 6 presents the visualization result. Specifically, Fig. 6(a) shows the preference of user u_{57} on the interacted items and clustered relations, where the blue, orange, green node denotes user, item and entity respectively. Fig. 6(b) shows the semantics of the relations in case study and the corresponding virtual relations. Fig. 6(c) show the exposure count of the virtual relations.

Fig. 6(a) exhibits that user's preference for the interacted items is different and i_{33} is more important to u_{57}'s preference modeling for it enjoys a higher weight. From Fig. 6(a) and Fig. 6(b), We observe that r_{15} and r_{30} are replaced by virtual relation v_0, for they are highly relevant and both related to *award*. And clearly, relations aligned with virtual relation v_2 is more related to *place*. The attention score in Fig. 6(b) is the weight in item embedding fusion step (equation (18)). It denotes user's attention on the virtual relations. From the attention scores, we can infer that user u_{57} cares more about the *award* than *place* information when choosing an item. Thus, it can be explained that *User u_{57} selects item i_{33} because it is involved with the award u_{57} cares for.* Fig. 6(c) shows that the exposure count of the virtual relations is comparable and we can conclude the relation long-tail issue is alleviated by introducing virtual relations.

5 RELATED WORK

As noted in the introduction, the mostly related work on RS with knowledge graph can be divided into three categories:

5.1 Embedding-based Methods

Embedding-based methods [1, 4, 20, 24, 26, 35, 37] mainly take advantage of the techniques in knowledge graph embedding (eg. transE [2]) to get item embedding which is used as pre-trained knowledge in subsequent recommendation task. Wang et al. [24] design a deep knowledge-aware network to mine latent knowledge-level relations in between news for generating news' representations. Zhang et al. [35] hire TransE on knowledge graph and feed the learned embedding into matrix factorization(MF). Wang et al. [26] propose a multi-task feature learning model based on a deep semantic matching framework to learn item and entity embedding by a cross&compress unit. Although these methods absorb the facts in KG as side information, they mainly focus on the first-order connectivity and ignore the global relationships of entities.

5.2 Path-based Methods

Path-based methods [5, 8, 12, 19, 42] sample the KG to explore a path between a target user and a candidate item for discovering some latent relations in between them. For example, Ma et al. [19] propose a joint learning framework to leverage associations between items in KG for delivering explainable recommendation. Fan et al. [8] present a metapath-guided embedding method to take full advantage of side information for well representing user intents. Zhao et al. [42] characterize the rich semantics of the heterogeneous types of entities by combining meta-graph with matrix factorization and factorization machine. Hu et al. [12] design a co-attention mechanism for mining the mutual effect between different metapaths. However, this kind of methods suffers from two main problems: (1)It needs professional knowledge and is weak in transfer ability. (2)It is labor-intensive and time-consuming when applying on KG of large scale.

5.3 GNN-based Methods

GNN-based methods [3, 22, 23, 25, 27, 29, 30, 32] propagate information through the topology of knowledge graph and aggregate the information from the neighbor node to update the representation of the center nodes. Wang et al. [23] propose an end-to-end framework Ripplenet to iteratively extend user's potential interests along connections in a KG in order to stimulating the propagation of user's preferences. KGAT [29] models the high-order interactions in KG explicitly based on attention mechanism and applies the same attention network to learn user preference. Cao et al. [3] incorporate text-based method and knowledge graph-based method to make full use of description information. Tang et al. [22] devise AKUPM which leverages a self-attention network to weight the importance of intra-entity-interaction and inter-entity-interaction for learning user representation. KGIN [30] as a state-of-the-art GNN-based recommendation model, encodes relation embedding into representation to preserve the semantic of long-range connectivity in KG.

GNN-based methods is really popular with researchers nowadays due to the excellent and promising performance. However, to

the best of our knowledge, current GNN-based methods directly apply GNN scheme on the origin KG which contains relations with similar function and may not suitable for recommendation tasks. Additionally, existing GNN-based methods aggregate all relational information into one embedding, which, we argue, will damage semantic information. In this paper, we propose to construct virtual relational knowledge graphs to preserve the semantic integrity and devise a local weighted smoothing (LWS) aiming at collecting side information from VRKGs to facilitate item representation for better recommendation performance.

6 CONCLUSION

Although knowledge graph is helpful for enriching item and user representation, we argue that it is not wise to directly exploit all relations of an origin KG without considering a particular downstream task. In this work, we have proposed a VRKG4Rec model to first construct VRKGs to learn a kind of virtual relational knowledge for item encoding. We have designed the LWS, a new graph neural model, for node encoding in a graph, which has been applied in each VRKG for item encoding. A fusion mechanism is used to learn final item representation. Experiments on two datasets have shown that the proposed VRKG4Rec outperforms the state-of-the-art methods. It is worth mentioning that we set the number of VRKGs manually in our model and it is labor-intensive and time-consuming to find the optimum number. Our future work shall investigate how to learn the optimal VRKG number automatically.

ACKNOWLEDGEMENTS

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 Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems* 26 (2013).

[3] Xianshuai Cao, Yuliang Shi, Han Yu, Jihu Wang, Xinjun Wang, Zhongmin Yan, and Zhiyong Chen. 2021. DEKR: Description Enhanced Knowledge Graph for Machine Learning Method Recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 203–212.

[4] 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 *The world wide web conference*. 151–161.

[5] Rose Catherine and William Cohen. 2016. Personalized recommendations using knowledge graphs: A probabilistic logic programming approach. In *Proceedings of the 10th ACM conference on recommender systems*. 325–332.

[6] Yu Chen, Ananya Subburathinam, Ching-Hua Chen, and Mohammed J Zaki. 2021. Personalized Food Recommendation as Constrained Question Answering over a Large-scale Food Knowledge Graph. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 544–552.

[7] Debashis Das, Laxman Sahoo, and Sujoy Datta. 2017. A survey on recommendation system. *International Journal of Computer Applications* 160, 7 (2017).

[8] Shaohua Fan, Junxiong Zhu, Xiaotian Han, Chuan Shi, Linmei Hu, Biyu Ma, and Yongliang Li. 2019. Metapath-guided heterogeneous graph neural network for intent recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2478–2486.

[9] László Grad-Gyenge, Attila Kiss, and Peter Filzmoser. 2017. Graph embedding based recommendation techniques on the knowledge graph. In *Adjunct publication of the 25th conference on user modeling, adaptation and personalization*. 354–359.

[10] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[11] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 355–364.

[12] 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 *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1531–1540.

[13] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 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.

[14] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.).

[15] Walid Krichene and Steffen Rendle. 2020. On Sampled Metrics for Item Recommendation. , 1748–1757 pages.

[16] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*. 2181–2187.

[17] Shenghao Liu, Bang Wang, and Minghua Xu. 2017. Event recommendation based on graph random walking and history preference reranking. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*. 861–864.

[18] Shenghao Liu, Bang Wang, Laurence T Yang, and S Yu Philip. 2021. HNF: Hybrid Neural Filtering Based on Centrality-Aware Random Walk for Personalized Recommendation. *IEEE Transactions on Network Science and Engineering* 9, 3 (2021), 1056–1066.

[19] Weizhi Ma, Min Zhang, Yue Cao, Woojeong Jin, Chenyang Wang, Yiqun Liu, Shaoping Ma, and Xiang Ren. 2019. Jointly learning explainable rules for recommendation with knowledge graph. In *The World Wide Web Conference*. 1210–1221.

[20] Enrico Palumbo, Giuseppe Rizzo, and Raphaël Troncy. 2017. Entity2rec: Learning user-item relatedness from knowledge graphs for top-n item recommendation. In *Proceedings of the eleventh ACM conference on recommender systems*. 32–36.

[21] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. Fast Context-Aware Recommendations with Factorization Machines. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 635–644.

[22] Xiaoli Tang, Tengyun Wang, Haizhi Yang, and Hengjie Song. 2019. AKUPM: Attention-enhanced knowledge-aware user preference model for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1891–1899.

[23] 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 *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 417–426.

[24] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. DKN: Deep knowledge-aware network for news recommendation. In *Proceedings of the 2018 world wide web conference*. 1835–1844.

[25] 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 *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 968–977.

[26] Hongwei Wang, Fuzheng Zhang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2019. Multi-task feature learning for knowledge graph enhanced recommendation. In *The World Wide Web Conference*. 2000–2010.

[27] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. 2019. Knowledge Graph Convolutional Networks for Recommender Systems. In *The World Wide Web Conference*. 3307–3313.

[28] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering* 29, 12 (2017), 2724–2743.

[29] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 950–958.

[30] Xiang Wang, Tinglin Huang, Dingxian Wang, Yancheng Yuan, Zhenguang Liu, Xiangnan He, and Tat-Seng Chua. 2021. Learning Intents behind Interactions with Knowledge Graph for Recommendation. In *Proceedings of the Web Conference*

2021. 878–887.

[31] Zhigang Wang, Juanzi Li, Zhiyuan Liu, and Jie Tang. 2016. Text-enhanced representation learning for knowledge base. In *Proceedings of International Joint Conference on Artificial Intelligent (IJCAI).* 4–17.

[32] Ze Wang, Guangyan Lin, Huobin Tan, Qinghong Chen, and Xiyang Liu. 2020. CKAN: Collaborative Knowledge-aware Attentive Network for Recommender Systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval.* 219–228.

[33] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. 2020. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys (CSUR)* (2020).

[34] Xuejiao Yang and Bang Wang. 2019. Local matrix approximation based on graph random walk. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval.* 1037–1040.

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

[36] Jiazheng Zhang and Bang Wang. 2022. Dismantling Complex Networks by a Neural Model Trained from Tiny Networks. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management.* 2559–2568.

[37] Yongfeng Zhang, Qingyao Ai, Xu Chen, and Pengfei Wang. 2018. Learning over knowledge-base embeddings for recommendation. *arXiv preprint arXiv:1803.06540* (2018).

[38] Zizhuo Zhang and Bang Wang. 2020. Learning sequential and general interests via a joint neural model for session-based recommendation. *Neurocomputing* 415 (2020), 165–173.

[39] Zizhuo Zhang and Bang Wang. 2021. Fusion of latent categorical prediction and sequential prediction for session-based recommendation. *Information Sciences* 569 (2021), 125–137.

[40] Zizhuo Zhang and Bang Wang. 2021. Graph Neighborhood Routing and Random Walk for Session-based Recommendation. In *2021 IEEE International Conference on Data Mining (ICDM).* IEEE, 1517–1522.

[41] Zizhuo Zhang and Bang Wang. 2022. Graph Spring Network and Informative Anchor Selection for Session-based Recommendation. *arXiv preprint arXiv:2202.09502* (2022).

[42] Huan Zhao, Quanming Yao, Jianda Li, Yangqiu Song, and Dik Lun Lee. 2017. Meta-graph based recommendation fusion over heterogeneous information networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining.* 635–644.

[43] Ding Zou, Wei Wei, Xian-Ling Mao, Ziyang Wang, Minghui Qiu, Feida Zhu, and Xin Cao. 2022. Multi-Level Cross-View Contrastive Learning for Knowledge-Aware Recommender System. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval.* 1358–1368.

[44] Ding Zou, Wei Wei, Ziyang Wang, Xian-Ling Mao, Feida Zhu, Rui Fang, and Dangyang Chen. 2022. Improving Knowledge-Aware Recommendation with Multi-Level Interactive Contrastive Learning. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management.*