Learning from Hierarchical Structure of Knowledge Graph for Recommendation

YINGRONG QIN, Tsinghua University, China
CHEN GAO*, Tsinghua University and Huawei Noah's Ark Lab, China
SHUANGQING WEI, Louisiana State University, The United States
YUE WANG, Tsinghua University, China
DEPENG JIN, Tsinghua University, China
JIAN YUAN, Tsinghua University, China
LIN ZHANG, Tsinghua Shenzhen International Graduate School, China
DONG LI, Huawei Noah's Ark Lab, China
JIANYE HAO, Huawei Noah's Ark Lab, China
YONG LI, Tsinghua University, China

Knowledge graphs (KGs) can help enhance recommendation, especially for the data-sparsity scenario with limited user-item interaction data. Due to the strong power of representation learning of graph neural networks (GNNs), recent works of KG-based recommendation deploy GNN models to learn from both knowledge graph and user-item bipartite interaction graph. However, these works have not well considered the *hierarchical structure* of knowledge graph, leading to sub-optimal results. Despite the benefit of hierarchical structure, leveraging it is challenging since the structure is always partly-observed. In this work, we first propose to reveal unknown hierarchical structures with a supervised signal detection method and then exploit the hierarchical structure with disentangling representation learning. We conduct experiments on two large-scale datasets, of which the results well verify the superiority and rationality of the proposed method. Further experiments of ablation study with respect to key model designs have demonstrated the effectiveness and rationality of our proposed model. The code is available at https://github.com/tsinghua-fib-lab/HIKE.

CCS Concepts: • **Information systems** → **Recommender systems**; • **Computing methodologies** → *Neural networks.*

Additional Key Words and Phrases: Personalized Recommendation, Knowledge Graph, Disentangled Representation Learning, Graph Neural Network

*The corresponding author

Authors' addresses: Yingrong Qin, Tsinghua University, China, qyr16@mails.tsinghua.edu.cn; Chen Gao, Tsinghua University and Huawei Noah's Ark Lab, China, chgao96@gmail.com; Shuangqing Wei, Louisiana State University, The United States, swei@lsu.edu; Yue Wang, Tsinghua University, China, wangyue@mail.tsinghua.edu.cn; Depeng Jin, Tsinghua University, China, jindp@tsinghua.edu.cn; Jian Yuan, Tsinghua University, China, jyuan@tsinghua.edu.cn; Lin Zhang, Tsinghua Shenzhen International Graduate School, China, linzhang@tsinghua.edu.cn; Dong Li, Huawei Noah's Ark Lab, China, lidong106@huawei.com; Jianye Hao, Huawei Noah's Ark Lab, China, haojianye@huawei.com; Yong Li, Tsinghua University, China, liyong07@tsinghua.edu.cn.

1046-8188/2023/7-ART $15.00
https://doi.org/10.1145/3595632

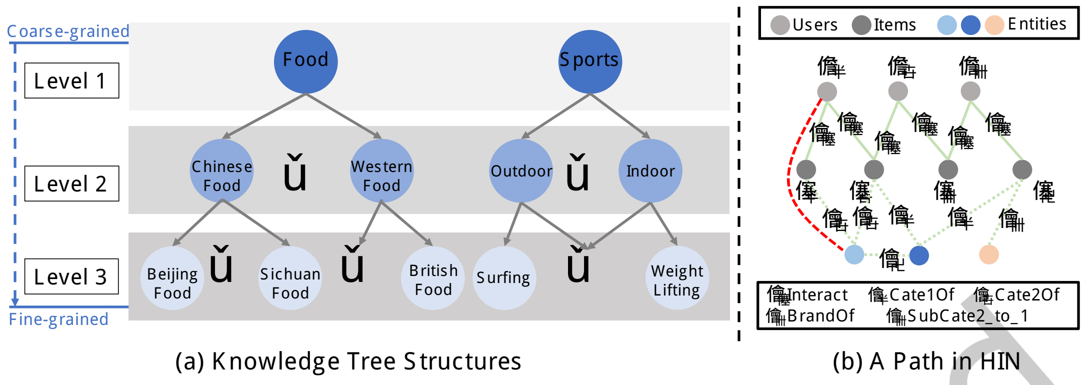

Fig. 1. The difference of term *hierarchy* in this paper and previous works. In Fig. 1 (a), the hierarchical knowledge tree structure (KTS) mentioned in this paper refers to structures that reveal domain knowledge from coarse-grained level to fine-grained level. In contrast, *hierarchy* used in previous works [10, 34, 50] refers to the connectivity between nodes of the same path in a heterogeneous information network (HIN). In Fig. 1 (b), we use a dashed red line to illustrate a path in HIN, which is totally different from the hierarchical structure in this paper.

1 INTRODUCTION

In the last decades, we have experienced the convenience of online services, including e-commerce, news, videos, etc., while facing the difficulty of information access due to the information explosion of the cyber world. Aiming at filtering valuable information for users as well as increasing profits for online platforms, recommender systems have appeared and have made a difference in many scenarios.

However, almost all recommendation scenarios suffer from the data-sparsity problem [6], which can cause great difficulty in capturing user preferences due to insufficient user-item interaction. To alleviate this problem, knowledge graphs (KGs) [3, 18, 22], which encode auxiliary information with the graph structure, have been incorporated into recommendation [4, 7, 31, 35, 36, 38, 45], named as KG-based recommendation. Some earlier works [1, 4, 45] first extract information from KG triplets, and then treat the distilled information as side information of items, which can help to obtain more representative embedding of users and items in the parameter optimization process and further benefit the downstream tasks.

Despite the effectiveness of these embedding-based approaches, they only leverage the first-order connectivity of KG. For the purpose of absorbing information from longer paths in KG, path-based KG-aware methods [5, 17, 27, 33, 40] have been proposed, in which predictions between users and items are made via estimating scores of paths connecting them. However, these methods highly rely on the quality of manually extracted paths. Lately, the paradigm of developing an end-to-end model based on graph neural networks (GNNs) has been widely adopted in [36, 38, 42, 46]. Combining GNNs with different information propagation mechanisms, these models usually have stronger prediction ability and interpretability.

In these various efforts in KG-based recommendation, an important property behind the knowledge graph is still less explored: The hierarchical knowledge tree structures in KG. Specifically, in a recommender system, there naturally exists hierarchical domain knowledge under different granularity. For example, for a given item in recommendation, its coarse-grained knowledge may be that *it is a restaurant* while fine-grained knowledge may be that *it is a French restaurant*. The coarse-to-fine-grained knowledge naturally forms hierarchical knowledge tree structures (KTSs), which is illustrated in Fig. 1 (a). Differently, some works may have used the term *hierarchy* to express the connectivity between nodes belonging to the same path in the heterogeneous information network

(HIN), such as the red dotted line illustrated in Fig. 1 (b). However, leveraging the hierarchical structure of the knowledge graph into recommendation is facing two critical challenges as follows.

- **The hierarchical structure can be incomplete**. Although some KTSs may have already been contained in the knowledge graph, there still exist unobserved KTSs due to the cost of constructing a knowledge graph. For instance, the correlation between *beer* and *diaper* has been observed in supermarkets since many young fathers would like to buy beers when they buy diapers for babies; however, *beer* and *diaper* belong to two KTSs, between which there exist no connections. These missing connections may worsen the quality of learned features and further affect the recommendation performance. Therefore, how to infer unknown structures based on the partly-observed one is challenging.
- **User preference learning from coarse-to-fine-grained levels hierarchically**. Since the hierarchical KTSs can reveal item functions in different granularity, capturing user preference at different granularities can probably help the recommender systems understand user demands better. However, it is hard to capture user interests from coarse-to-fine-grained levels hierarchically due to users' direct connection with items rather than nodes of KTSs in HIN. Therefore, how to leverage the hierarchical KTSs to capture user preferences at different granularity is another critical challenge.

To address these two challenges mentioned above, in this paper, we proposed a solution named HIKE (short for learning **HI**erarchical structure of **K**nowledge Graph for r**E**commendation). Specifically, we first propose to implement relation-aware representation learning in the original space, as well as a supervision signal detection module. We then propose a disentangled representation learning method, which can assign multiple embedding for items in different disentangled spaces, which can reflect item functions and user preferences at different granularity. In addition, we propose a compound loss function to benefit the representation learning in the original space from the disentangled latent spaces.

To summarize, the contribution of this paper can be summarized as follows.

- We take an early step of fully leveraging hierarchical knowledge tree structures to enhance the predictive ability of KG-based recommender systems.
- We propose a supervised signal detection method with knowledge tree structures and a disentangling representation learning method, accompanied by an additional supervision loss.
- We conduct experiments on two large-scale real-world datasets, and the results have shown the superior performance of our method compared with state-of-the-art approaches. Further ablation studies on important designs of HIKE have demonstrated its effectiveness.

The remaining of this paper is organized as follows. We first formulate the studied problem in Section 2. We then present our proposed method in Section 3. Extensive experiments are reported in Section 4, after which we review important related works in Section 5. Last, we conclude the paper and discuss the future works in Section 6.

2 PRELIMINARIES AND PROBLEM FORMULATION

In this section, we formulate the problem solved in this paper by first describing input data and an important concept. Then, we formulate the problem for a clearer statement. Besides, we provide Table 1 to explain the meaning of frequently used notations in this paper.

2.1 Knowledge Graph

A KG is usually composed of enormous KG triplets in the format of (h, r, t), where h, t indicate head entity and tail entity, and r indicates the relation which connects them. Since KG can store real-world facts, common sense, domain knowledge, etc., it is usually powerful and can help to enhance the explainability of recommender systems. In this paper, we use the urban KG constructed in [23] to supplement item side information. Specifically,

Table 1. Frequently used notations in this paper.

Notation	Meaning
$\mathcal{G}, \mathcal{V}, \mathcal{E}$	The knowledge graph, its vertex set, and its node set
$\mathcal{G}', \mathcal{V}', \mathcal{E}'$	The heterogeneous graph consisting of KG and the user-item bipartite graph, its vertex set, and its node set
$\mathcal{U}, \mathcal{I}$	The user set and the item set
$u, i, N(u), N(i)$	User u, item i, neighborhood of user u and neighborhood of item i
$L, \mathbf{e}_u^{(l)}, \mathbf{e}_i^{(l)}$	The number of layers of the GNN model, user embedding in layer l and item embedding in layer l
$C_{k,j}, \mathbf{v}_{k,j}$	The jth cluster at level k and the corresponding cluster center
$\mathbf{e}_u, \mathbf{e}_u^k$	The original embedding of user u obtained from the GNN model, and its disentangled component at level k
$\mathbf{e}_i, \mathbf{e}_i^k$	The original embedding of item i obtained from the GNN model, and its disentangled component at level k
$\mathbf{e}_u^{C_{k,j}}, \mathbf{e}_i^{C_{k,j}}$	Sub-vector of user u and item i obtained after projecting to cluster $C_{k,j}$
I_u^+, I_u^-	The positive/negative set containing all items user u has/hasn't engaged with
$\mathcal{L}, \mathcal{L}^o, \mathcal{L}^{dis}$	The total loss, the loss of the original space, and the loss of the disentangled space
$\mathcal{L}_k$	The disentangled loss at level k
$s_{u,i}, s_{u,C_{k,j}}$	Similarity score between user u and item i, similarity score between user u and cluster $C_{k,j}$

let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ indicate the KG, where $\mathcal{E}$ denotes the edge set containing all relations, and $\mathcal{V}$ indicates the nodes set consisting of both head entities and tail entities of all KG triplets. In Fig. 2, we provide an illustration of the deployed urban KG for a better explanation. Besides, following [10, 38], we also adopt inverse relations to form new triplets. For instance, the KG stores the first-level categorical information of item *Quanjude* via the relation *IsCate1Of* in (*Food*, *IsCate1Of*, *Quanjude*), and the corresponding inverse relation *Cate1* is related with the triplet (*Quanjude*, *Cate1*, *Food*).

2.2 Bipartite Interaction Graph

To utilize the superior message-passing strategy of GNN when generating user embedding, we build the user-item bipartite interaction graph. As Fig. 2 illustrates, edges between users and items exist if users have interacted with those items historically. Let $\mathcal{G}_B$ indicate the bipartite interaction graph, and let $\mathcal{G}_B$ indicate the vertex set, including all users and all items, while edges in $\mathcal{G}_B$ indicate the interaction between users and items.

2.3 Knowledge Tree Structure

As illustrated in Fig. 1(a), the knowledge tree structures (KTSs) mentioned in this paper indicate the tree structures that reflect domain knowledge. More specifically, for a k-level KTS, domain knowledge varies from coarse-grained to fine-grained when its level changes from level 1 to level k. For instance, as for item *KFC*, it belongs to *Food* at level 1, and belongs to *American Food* at level 3. Thus, each level of the correlated KTSs provides side information about item *KFC* from the perspective of domain knowledge. Let $c_{1,j}$ indicate the j-th node of the k-level KTS at the first level. Similarly, $c_{2,j}$ and $c_{k,j}$ indicate the jth node of the second level and the kth level, respectively. In this paper, we highlight the importance of KTSs in KG because the scarce user-item interaction data increases the difficulty of obtaining high-quality feature vectors, while combining domain knowledge reflected by KTSs at all levels can alleviate the data sparsity problem.

2.4 Problem Formulation

As mentioned above, KTSs reflect domain knowledge from the coarse-grained level to the fine-grained level. To explore how recommender systems can benefit from KTSs, we aim to find a hierarchy-based KG-enhanced recommendation approach. Differently, the word *hierarchy* in previous works [10, 34, 50] refers to the connectivity

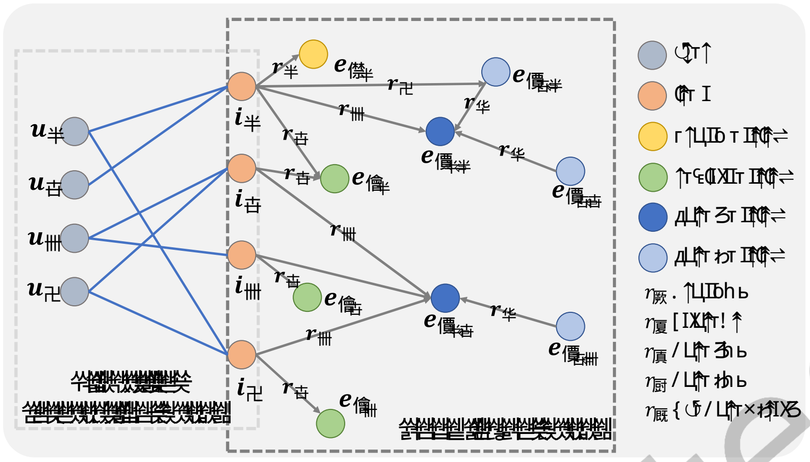

Fig. 2. Illustration of the user-item bipartite interaction graph and knowledge graph used in this paper (Best viewed in color). As illustrated, the bipartite interaction graph shares the same items with the KG. Since items bridge the user-item bipartite interaction graph and the KG, information propagation on the KG can also benefit the representation learning of users on the bipartite interaction graph.

between nodes of the same path in heterogeneous information networks (HIN). We illustrate this major difference in Fig. 1.

Given a dataset with M users and N items, let $\mathcal{U}$ indicate the user set and $\mathcal{I}$ indicate the item set. Then, let $I_u^+ = \{i_1, i_2, \cdots, i_n\}$ denote user u's positive item set, in which all items have been engaged with user u. Similarly, I_u^- is used to indicate user u's negative item set, consisting of items that user u has never engaged with. We extend knowledge graph $\mathcal{G}$ to a heterogeneous graph $\mathcal{G}'$ by adopting interaction between users and items as a special relation r_i. In other words, the interaction between user u and item i is represented as triplets (u, r_i, i). Therefore, the heterogeneous graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$, where $\mathcal{V}' = \mathcal{V} \cup \mathcal{U}$ and $\mathcal{E}' = \mathcal{E} \cup \{r_i\}$. Now, the problem to solve is how to leverage the known KTSs in KG to improve KG-enhanced recommendation performance, which utilizes information in HIN $\mathcal{G}'$.
Input: $\mathcal{G}'$ and KTSs in KG.
Output: Find a function f, which estimates how likely user u will engage with item i based on $\mathcal{G}'$ and KTSs.

3 METHODOLOGY

For the purpose of understanding user interests and item functions at different granularity, we propose HIKE, an end-to-end GNN-based model, which incorporates relation-aware knowledge graph embedding (KGE) and disentangles user interests from coarse-grained granularity to fine-grained granularity to improve recommendation performance. There are four important components in HIKE:

- **Relation-aware Information Propagation Module**. For the purpose of obtaining high-quality user embedding and item embedding, we propose this module which passes the message over the extended heterogeneous graph $\mathcal{G}'$ in a relation-aware way.
- **Supervision Signal Detection Module**. We extract valuable supervision signals from KTSs through this proposed module, which clusters entities at the same level among different KTSs. Later, the output clusters

will be used as supervision signals to guide the representation learning process and help to alleviate the data sparsity problem by leveraging the domain knowledge in KTSs.
- **Disentangled Embedding Learning Module**. In this module, we learn disentangled user embedding and disentangled item embedding by leveraging the original embedding obtained from the relation-aware information propagation module and the supervision signals from the supervision signal detection module.
- **Compound Objective Function Module**. Since both disentangled embedding and original embedding are learned in our model, we design a compound objective function that leverages the optimization in the disentangled spaces to assist the optimization in the original space.

3.1 Relation-aware Information Propagation

In KG-enhanced recommender systems, the implementation of KGE is critical. Specifically, since KG can help to obtain high-quality user embedding and item embedding, the way of information extraction from KG can make a big difference when choosing different information propagation mechanisms. Before deciding how to implement KGE, we must take the following three aspects into consideration:

- how to deal with different relations during information propagation;
- how to aggregate passing messages from different neighbors;
- how to generate user embedding by utilizing KG.

Considering all these three aspects, we design a relation-aware information propagation mechanism, which strengthens the influence of different relations via relation-aware filtering during message passing. Besides, after the relation-aware filtering, messages passed from different neighbors will be treated equally. Meanwhile, we execute message passing on KG and user-item bipartite graph recursively to benefit the user representation learning process.

3.1.1 ***Information Propagation over KG***. Information propagation over KG can help to obtain item side information, which is stored in KG triplets. Since items bridge the bipartite graph and the KG (illustrated in Figure 2), the results of KGE will enhance the quality of user embedding implicitly and item embedding explicitly. As illustrated in Figure 2, there exist different paths between the same node pairs. For instance, path A $i_1 \xrightarrow{r_3} e_{c_{1,1}}$ and path B $i_1 \xrightarrow{r_4} e_{c_{2,1}} \xrightarrow{r_5} e_{c_{1,1}}$ connect the same node pair $(i_1, e_{c_{1,1}})$ on KG. Let $N(i_1)$ indicate item i_1's directly connected neighborhood. Then, the passing message from $N(i_1)$ to i_1 is computed as:

$$\mathbf{m}_{i_1 \leftarrow N(i_1)} = \mathrm{Mean}\left(\mathbf{e}_r \odot \mathbf{e}\right), \quad \forall e \in N(i_1), \tag{1}$$

where $\odot$ indicates element-wise product, and entities in $N(i_1)$ exist in i_1-relative KG triplets such as (i_1, e_r, e) or (e, e_r, i_1). In this way, edges that indicate different relations are used as filters, and message from different neighbors is propagating in a relation-aware method. Thus, the influence of different relations will be distinguished during the information propagation process. In addition, as mentioned in Section 2, HIKE also adopts the inverse relation when generating KG triplets, following [10, 36, 38].

In a L-layer HIKE model, information propagation is executed inside each layer. Embedding of i_1 in layer $l+1$ is computed from embedding of layer l as follows,

$$\mathbf{e}_{i_1}^{(l+1)} = \mathbf{e}_{i_1}^{(l)} + \mathrm{Mean}\left(\mathbf{e}_r \odot \mathbf{e}_s^{(l)}\right), \quad \forall s \in N(i_1),\ l = 0, 1, \cdots, L-1, \tag{2}$$

where the superscripts are used to distinguish the embedding of different layers. Therefore, by stacking layers, message passing from multi-hop neighbors, such as $\mathbf{m}_{i_1 \leftarrow e_{c_{1,1}}}$, will be achieved via $\mathbf{m}_{i_1 \leftarrow e_{c_{2,1}}}$ implicitly. In this way, KGE in HIKE leverages both first-order connectivities between directly connected neighbors and high-order connectivity between multi-hop neighbors. Moreover, embedding with superscript (0) is generated by random initialization in the beginning.

Thus, after computing $\mathbf{e}_{i_1}^{(0)}, \cdots, \mathbf{e}_{i_1}^{(l)}, \cdots, \mathbf{e}_{i_1}^{(L)}$ in a L-layer HIKE model, the final embedding of entity i_1 in KG is computed as:

$$\mathbf{e}_{i_1} = \frac{1}{L+1}\left(\mathbf{e}_{i_1}^{(0)} + \mathbf{e}_{i_1}^{(l)} + \cdots + \mathbf{e}_{i_1}^{(L)}\right). \tag{3}$$

Additionally, relation embedding $\mathbf{e}_r$ is first initialized with small values. During training, it will be optimized via gradient backward. Note that the embedding of entities in KG will be updated in GNN layer-by-layer, while the embedding of relations is the same across all GNN layers.

Since inverse paths and inverse relations have been adopted, the embedding of all KG entities, such as $\mathbf{e}_{\mathbf{c}_{1,1}}, \mathbf{e}_{\mathbf{c}_{2,1}}$, etc., can be computed similarly. In addition, the embedding of all KG relations will be initialized randomly and optimized during training.

Besides, attentive modules, such as the attentive combination of KG relations used in [38], and the attentive weights used in [36], have not been deployed in our approach. This is reasonable because we can regard the attention mechanism as an entangled operation, which attributes different weights to the passing messages from different paths or different nodes. These attentive weights are obtained automatically during training, and the combination is executed according to inexplicable reasons. Differently, HIKE disentangles item functions and user preferences with the help of KTSs. We will explain the detail in Section 3.3, in which explicable disentanglement is implemented with the help of domain knowledge in KTSs. Further experimental results in Section 4 have also demonstrated the effectiveness of the disentangled design.

3.1.2 ***Information Propagation over the Bipartite Interaction Graph***. As mentioned above, information propagation over the KG can help to learn more representative item embedding. Similarly, we implement user representation learning by passing messages on the user-item bipartite graph. Compared with the KG, in which multiple types of relations exist, there is only one relation, interaction, in the user-item bipartite graph (illustrated in Fig. 2). Note that items bridge the KG and the bipartite graph. Therefore, after information propagation is finished on KG, messages will be passed between items and users on the bipartite graph. That is, the obtained item embedding (such as $\mathbf{e}_{i_1}$ obtained following equation (3)) will be utilized to generate user embedding. Thus, the item-side information stored in KG triplets can help to obtain more representative user embedding implicitly. In addition, considering that there is only one edge type in the user-item bipartite graph, information propagation between users and items is carried out without strengthening the edge influence. Thus, in the l^{th} layer of HIKE, it is computed as follows,

$$\mathbf{e}_u^{(l)} = \text{Mean}(\mathbf{e}_i^{(l-1)}), \quad i \in N(u), \tag{4}$$

$$\mathbf{e}_i^{(l)} = \text{Mean}(\mathbf{e}_u^{(l-1)}), \; u \in N(i), \tag{5}$$

where $N(u)$ and $N(i)$ indicate directly connected items and users on the bipartite graph of user u and item i, respectively. Similar to equation (3), the embedding of a specific user u_1 user is computed as:

$$\mathbf{e}_{u_1} = \frac{1}{L+1}\left(\mathbf{e}_{u_1}^{(0)} + \mathbf{e}_{u_1}^{(l)} + \cdots + \mathbf{e}_{u_1}^{(L)}\right). \tag{6}$$

Note that information propagation on the KG and propagation on the bipartite graph are executed alternatively during training.

3.2 Supervision Signal Detection via KTSs

Supposing that there is an ideal situation in which sufficient data and complete prior knowledge are available, then representation learning will be easy and capable of capturing real user interests. However, the real situation often suffers from data sparsity problems as well as incomplete domain knowledge. Therefore, how to mine unknown knowledge from known information is critical and challenging. In this paper, we propose to leverage

KTSs to find supervision signals, according to which the representation learning process can benefit more from the domain knowledge. Two main advantages of this design are listed as follows,

- **Emphasizing the impact of known domain knowledge.** Labels which reflect domain knowledge are usually sparse and expensive in most datasets. Therefore, how to make the best use of available labels is significant. Compared with labels extracted from review data etc., the KTSs used in this paper are more common in recommendation (e.g., the movie classification information in e-commerce platforms). However, previous works achieve KGE by focusing on KG triplets or KG paths, which can lead to underestimation of the fine-to-coarse-grained hierarchical knowledge structures. In this paper, mining supervision signals from KTSs can help us understand user interests and item functions at different levels, which increases the robustness and avoids overfitting to some extent during representation learning.
- **Enhancing the similarity beyond KTSs.** Available domain knowledge is incomplete in most cases. Optimizing the desired user embedding and item embedding in perfect alignment with known domain knowledge can lead to over-fitting easily. Thus, the downstream tasks can only achieve sub-optimal performance. To avoid the problem mentioned above, we form supervision signals from KTSs instead of using KTSs directly. Further experiments have demonstrated the rationality of these supervision signals.

As illustrated in Fig. 3, supposing that several three-level KTSs (consisting of category information) are available, we use $c_{1,\cdot}$ to donate nodes of the coarse-grained classified information at the first level. Similarly, $c_{2,\cdot}$ and $c_{3,\cdot}$ indicate nodes of the second level and the third level, respectively. Later, clustering is executed among nodes of the same levels, respectively.

Once embedding of nodes in KTSs is accessible, we can obtain clusters at different levels by applying the K-means algorithm [28]. Let $C_{k,j}, v_{k,j}$ indicate the jth cluster and its cluster center at the kth level as:

$$C_{k,1}, C_{k,\cdot}, C_{k,n_k} = \text{K-means}(\mathbf{e}_{c_{k,1}}, \mathbf{e}_{c_{k,2}}, \cdots), \tag{7}$$

where n_k indicates the cluster number at level k, and $\mathbf{e}_{c_{k,\cdot}}$ indicates embedding of KG entity $c_{k,\cdot}$ at level 2 in KTSs. Then, the representation of the corresponding cluster center is formulated as:

$$\mathbf{v}_{k,j} = Mean(\mathbf{e}_{c_{k,\cdot}} | \forall e_{c_{k,\cdot}} \in C_{k,j}). \tag{8}$$

Since similar categories at the same level can be clustered into the same cluster, such as *Tea* and *Coffee*, the results of clustering will be used as supervision signals during representation learning. More specifically, before clustering, we only know a specific item i belongs to category *Coffee* at level 3. After clustering, we now know Cate3 of item i, *Coffee* belongs to some cluster $C_{3,\cdot}$, in which there are similar entities such as *Tea* at level 3. Therefore, during representation learning, we first try to ensure $\mathbf{e}_i$ near to $\mathbf{v}_{3,\cdot}$. In this way, the obtained embedding can be more robust and can help the recommender systems find user interests at different levels with respect to the clusters at all levels. Besides, by combining the clustering results at all levels, we can reduce the noise from inappropriate results of entity clustering.

Now, the only remaining problem is how to get the embedding of nodes in KTSs. Generally, any KGE approach can generate the embedding of entities in KTSs. During experiments, we observed positive relation between model performance and the quality of KGE. In this paper, we use pre-trained KGE results from KGIN [38]. Moreover, members of each cluster at all levels are fixed during model training. Note that no pre-trained embedding will be loaded into our model, but it is employed as supervision signals to generate disentangled user embedding and item embedding. We will explain the detail in the next part.

Particularly, when remarkable semantic gaps exist in the meanings at the coarse-grained level, such as *Computer* and *Food*, we will omit the entity clustering process and use the embedding of these nodes as cluster centers directly.

Note that nodes in the same cluster might have different parent nodes. For instance, the leaf node *Dumpling* of *Chinese Food* and the leaf node *Hamburger* might be found in the same cluster, which may contain other fast

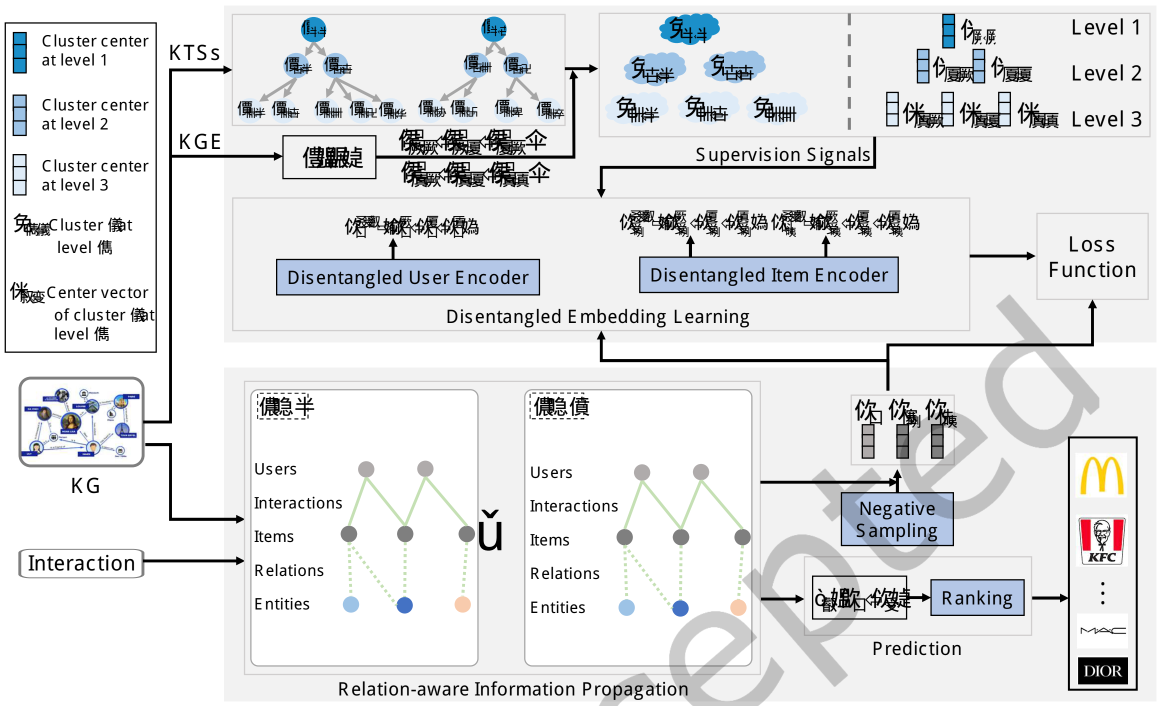
Cluster center at level 1
Cluster center at level 2
Cluster center at level 3
Center vector
KTSs
KGE
KG
Interaction
Level 1
Level 2
Level 3
Supervision Signals
Disentangled User Encoder
Disentangled Item Encoder
Disentangled Embedding Learning
Loss Function
Users
Interactions
Items
Relations
Entities
Negative Sampling
Ranking
Prediction
Relation-aware Information Propagation

Fig. 3. Framework of HIKE (Best viewed in color). Model inputs are interaction data and KG (consisting KTSs). First, relation-aware information propagation is executed within a GNN model. Meanwhile, supervision signals are detected leveraging KTSs. Then, HIKE generates disentangled user embedding and item embedding utilizing the output of the GNN model and the supervision signals. Besides, a compound objective function is employed, according to which the losses are computed both in the original space and the disentangled space. Representations of items and users will be optimized based on the gradients backward from the compound loss function. Finally, personalized recommendations will be implemented by ranking the matching scores between users and items in the prediction period.

foods such as *Pizza*. Therefore, clusters obtained via KTSs can be regarded as supplementary supervision signals and will benefit representation learning during optimization.

3.3 Disentangled Embedding Learning

Users and items define each other via interaction data. Specifically, the similarity between users is usually computed by the co-interact items in traditional collaborative filtering recommendation approaches. Likewise, the similarity between items can be defined by users. Recently, with the help of KG, item-side information can be explored and exploited to obtain high-quality user embedding and item embedding. Despite the effectiveness of existing approaches, we find that the effects of hierarchical KTSs could be underestimated. Most importantly, previous works ignore the information hidden in the hierarchical structures. In Figure 4(a), we illustrate an object consisting of three cylinders, together with its three-view drawing. In cases where information about the object is unknown or incomplete, its three-view drawing can help us recover it. Similarly, the motivation of computing disentangled components of items and users can help to obtain high-quality embedding in the original space. To be specific, interactions between users and items are usually sparse, which makes learning high-quality representations challenging. Facing the challenge mentioned above, we first leverage the domain knowledge in

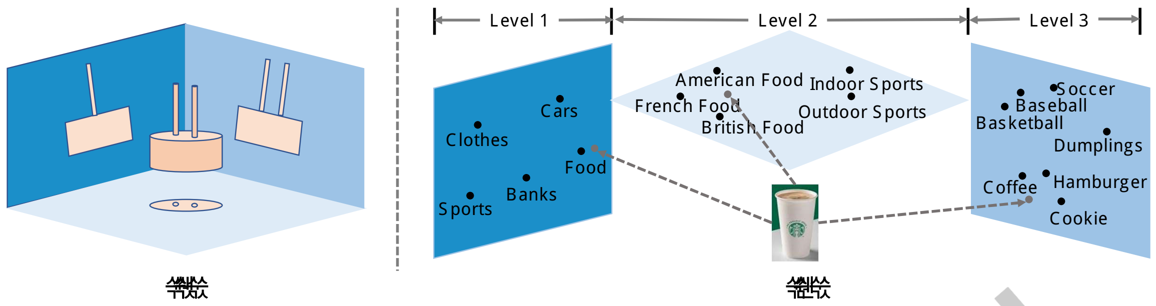

Fig. 4. A toy model to illustrate the effects of disentangled representation learning (Best viewed in color). In Fig. 4 (a), we illustrate the three-view drawing of an object, consisting of three cylinders. In circumstances when the object is unknown, its three-view drawing can help us recover it. It is similar to representation learning. Given sufficient data, accurate and explicable representations of users and items can be guaranteed easily. However, the observed data in the real world can always be sparse, which increases the difficulty of obtaining high-quality embedding greatly. To cope with this problem, we project each item to domain knowledge according to KTSs from the coarse-grained level to the fine-grained level. In Fig. 4 (b), we use gray dots to denote the disentangled components in the projection space. Since the cluster of each item's categories is fixed, HIKE tries to keep the disentangled components near their corresponding cluster centers during training.

KTSs to form three sub-spaces. Then, we project each item to these sub-spaces, and similarity retaining in these sub-spaces is highlighted. Ultimately, we find that the quality of embedding learning in the original space can be enhanced via embedding learning in the sub-spaces, which helps to alleviate the data sparsity problem. We will explain how to obtain disentangled embedding of users and items in this Section as follows.

3.3.1 ***Disentangled item encoder.*** KGE can be regarded as a process of information fusion, during which item embedding is obtained by aggregating information from KTSs-relevant knowledge (e.g., nodes at all levels) and KTSs-irrelevant knowledge (brand, etc.). For the purpose of guiding the representation learning with known domain knowledge, we disentangle item embedding to all levels with respect to the KTSs. Briefly, a disentangled item encoder is designed, whose output is feature vectors that reflect item functions at each level from the perspective of KTSs. Taking item i_1 as an example, we first project $\mathbf{e}_{i_1}$ to each level via the associated category entity $e_{c_{k,1}}$ as follow,

$$\hat{\mathbf{e}}_{i_1}^{k} = \sigma\left(\mathbf{e}_{i_1} \odot \mathbf{e}_{c_{k,1}}\right) \odot \mathbf{e}_{i_1} + \mathbf{e}_{i_1}, \quad k = 1, 2, 3, \tag{9}$$

where σ indicates the *sigmoid* function, $\mathbf{e}_{c_{k,1}}$ indicates the category entity that connects with item i_1 at level k. The operation in equation (9) can be treated as a filter, which amplifies relation information at the corresponding dimensions. After passing this filter, dimensions close to entity $e_{c_{k,1}}$ at level k will be strengthened by the non-linear transformation.

Then, the disentangled component $\mathbf{e}_{i_1}^{k}$ of i_1 at level k will be generated by two steps. Firstly, we project $\hat{\mathbf{e}}_{i_1}^{k}$ to all cluster centers $\mathbf{v}_{k,\cdot}$ at level k as follows,

$$\mathbf{e}_{i_1}^{C_{k,j}} = \hat{\mathbf{e}}_{i_1}^{k} \odot \mathbf{v}_{k,j}, \quad k = 1, 2, 3, \;\; j = 1, \cdots, n_k, \tag{10}$$

where $\mathbf{e}_{i_1}^{C_{k,j}}$ denotes the projected component of i_1 to cluster $C_{k,j}$ at the k-th level, and n_k indicates the cluster number of the kth level. In other words, cluster centers $\mathbf{v}_{k,1}, \mathbf{v}_{k,2}, \cdots, \mathbf{v}_{k,n_k}$, which are obtained in Section 3.2, are utilized as a group of bases. We think the projection vector $\mathbf{e}_{i_1}^{C_{k,j}}$ reflects the functional correlation between item

i_1 and cluster $C_{k,j}$ to some extent, and HIKE is proposed to find the true correlation between i_1 and all clusters at all levels during training.

After obtaining $\{\mathbf{e}_{i_1}^{C_{k,1}}, \mathbf{e}_{i_1}^{C_{k,2}}, \cdots, \mathbf{e}_{i_1}^{C_{k,n_k}}\}$ at each level, then, the disentangled components of i_1 *w.r.t* three-level KTSs are computed as:

$$\mathbf{e}_{i_1}^{\text{dis}} = \left(\left\{ \mathbf{e}_{i_1}^{C_{1,1}}, \cdots, \mathbf{e}_{i_1}^{C_{1,n_1}} \right\}, \left\{ \mathbf{e}_{i_1}^{C_{2,1}}, \cdots, \mathbf{e}_{i_1}^{C_{2,n_2}} \right\}, \left\{ \mathbf{e}_{i_1}^{C_{3,1}}, \cdots, \mathbf{e}_{i_1}^{C_{3,n_3}} \right\} \right). \tag{11}$$

During the disentangled representation learning process, we will not concatenate feature vectors generated from different clusters or combine them via functions to aggregate information immediately. Every single component at all levels will benefit due to the optimization process by computing pairwise loss with its corresponding component non-linearly, respectively.

Besides, for simplicity, we represent the disentangled components of item i_1 in the following part as:

$$\mathbf{e}_{i_1}^{dis} = \left(\mathbf{e}_{i_1}^1, \mathbf{e}_{i_1}^2, \mathbf{e}_{i_1}^3 \right), \tag{12}$$

where $\mathbf{e}_{i_1}^k$ actually indicates a group of i_1's disentangled components at level k corresponding to all clusters $C_{k,j}$ at level k.

The obtained disentangled components in $\mathbf{e}_{i_1}^{dis}$ can help to reveal user interests from the coarse-grained level to the fine-grained level, which will be demonstrated with further experiments in Section 4.5.

3.3.2 ***Disentangled user encoder.*** Similarly, disentangled user embedding is generated via a disentangled user encoder. Considering the fact that user-side information, such as user hobbies, category preferences, etc., is hard to collect in practice, the disentangled user encoder is designed by directly projecting user embedding to obtained cluster centers at each level. To be specific, for user u_1, the disentangled component $\mathbf{e}_{u_1}^k$ at level k is computed based on $\mathbf{e}_{u_1}^{C_{k,j}}$, which is computed as:

$$\mathbf{e}_{u_1}^{C^{k,j}} = \mathbf{e}_{u_1} \odot \mathbf{v_{k,j}}, \quad k = 1, 2, 3, j = 1, \cdots, n_k. \tag{13}$$

Similar to equation (12), we use the following equation to indicate disentangled components of user u_1 at three levels as follows:

$$\mathbf{e}_{u_1}^{dis} = \left(\mathbf{e}_{u_1}^1, \mathbf{e}_{u_1}^2, \mathbf{e}_{u_1}^3 \right), \tag{14}$$

where the disentangled component $\mathbf{e}_{u_1}^k$ at level k is:

$$\mathbf{e}_{u_1}^k = \left\{ \mathbf{e}_{u_1}^{C_{k,1}}, \cdots, \mathbf{e}_{u_1}^{C_{k,n_k}} \right\}, \quad k = 1, 2, 3. \tag{15}$$

Once obtaining the disentangled components of users and items, we'll show how these disentangled components can help representation learning in the original space in the following part.

3.4 Compound Objective Function

To optimize the proposed model both in the disentangled embedding spaces and in the original space, we propose a compound objective function. Utilizing this specially designed objective function, the embedding of the disentangled spaces will help to optimize the embedding of the original space.

3.4.1 ***Optimization.*** During the optimization process, we use the widely adopted Bayesian personalized ranking (BPR) loss [32] to compute losses in a pairwise way. Specifically, for each positive user-item pair (u, i_+), we randomly sample a negative item i_- from negative set I_u^-, where I_u^+/I_u^- indicates item set that user u has/hasn't interacted with historically. Therefore, let $O = \{(u, i_+, i_-)|u \in \mathcal{U}, i_+ \in I_U^+, i_- \in I_u^-\}$ denote the triplet set, in which a triplet consists of an already-interacted user-item pair and a never-engaged item from random negative sampling. Then, given O, BPR loss is computed as follows,

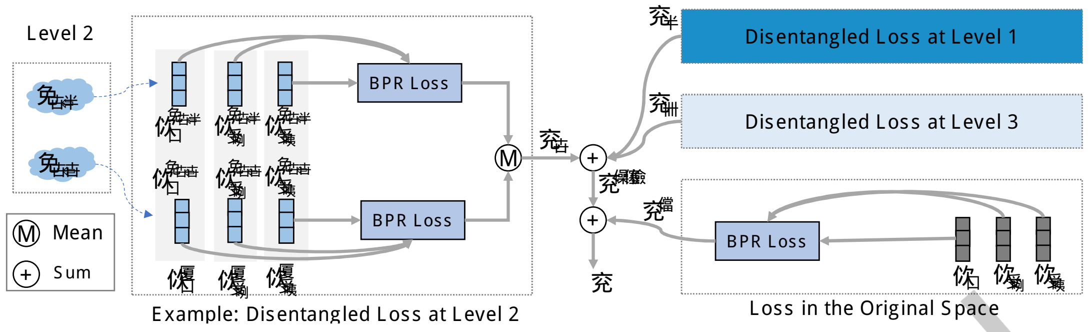
Level 2
BPR Loss
BPR Loss
Disentangled Loss at Level 1
Disentangled Loss at Level 3
BPR Loss
Mean
Sum
Example: Disentangled Loss at Level 2
Loss in the Original Space

Fig. 5. Loss computation in a disentangled way. The proposed compound objective function computes loss both in the original space and in the disentangled spaces at all levels. Specifically, taking level 2 as an example, we illustrate how disentangled loss is obtained in the left half of the figure. Similar operations are executed at level 1 and level 3.

$$\mathcal{L}_{\text{BPR}} = - \sum_{(u,i_+,i_-)\in O} \ln\sigma(\mathbf{e}_u^T\mathbf{e}_{i_+} - \mathbf{e}_u^T\mathbf{e}_{i_-}). \tag{16}$$

Actually, gradients propagating backward according to BPR loss will push user u and positive item i_+ closer in the embedding space while pulling away negative item i_- from user u. In this paper, we design a compound loss function, which leverages the optimization in disentangled spaces at all levels to help the optimization in the original space. To be specific, as illustrated in Figure 5, the total disentangled loss $\mathcal{L}^{dis}$ is computed of the disentangled loss $\mathcal{L}_k$ at all levels as follow,

$$\mathcal{L}^{dis} = \mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_3, \tag{17}$$

where the disentangled loss $\mathcal{L}_k$ is computed as:

$$\mathcal{L}_k^{dis} = -\frac{1}{n_k} \sum_{(u,i_+,i_-)\in O} \ln\sigma({\mathbf{e}_u^{C_{k,j}}}^T \mathbf{e}_{i+}^{C_{k,j}} - {\mathbf{e}_u^{C_{k,j}}}^T \mathbf{e}_{i-}^{C_{k,j}}), \quad k = 1, 2, 3, \quad j = 1, \cdots, n_k. \tag{18}$$

In other words, the pairwise BPR loss is computed among the disentangled components, which correspond to the same cluster of triplet (u, i_+, i_-), respectively. For the purpose of distinguishing losses computed in the disentangled spaces and the original space, we replace $\mathcal{L}_{BPR}$ with $\mathcal{L}^o$ when referring to losses in the original space.

Therefore, the proposed compound loss $\mathcal{L}$ of HIKE is computed as:

$$\mathcal{L} = \mathcal{L}^o + \mathcal{L}^{dis} + \lambda||\Theta||_2^2, \tag{19}$$

where $\Theta = \{\mathbf{e}_u, \mathbf{e}_s, \mathbf{e}_r | u \in \mathcal{U}, s \in \mathcal{V}, e_r \in \mathcal{E}\}$. Note that the $\mathcal{V}$ and $\mathcal{E}$ here are the node set and edge set of the knowledge graph, including all items. Therefore, the embedding of users and items can be optimized by minimizing loss $\mathcal{L}$.

3.4.2 ***Prediction.*** As illustrated in Fig. 3, personalized recommendation content generation is based on the representations of users and items. To be specific, we explain how candidate set T_u^n of length n is generated for a specific user u as follows. Firstly, the estimation score s_{ui} will be computed between user u and item $i, \forall i \in I_u^-$. Following [10, 36, 38], we calculate the inner product between $\mathbf{e}_u$ and the embedding of all his/her un-interacted items as follows:

Table 2. Statistics of Beijing Dataset and Shanghai Dataset

Dataset	User-Item Interaction			Knowledge Graph		
	#Users	#Items	#Interactions	#Entities	#Relations	#Triplets
Beijing	29,992	27,685	219,402	31,900	16	366,730
Shanghai	25,819	40,529	198,679	44,910	16	538,356

$$s_{u,i} = \mathbf{e}_u^T \mathbf{e}_i, \quad \forall u \in \mathcal{U}, \forall i \in \mathcal{I}. \tag{20}$$

Then, we rank all the calculated scores, and C_u is obtained by taking out the top n items in order.

3.4.3 ***Time Complexity Analysis.*** The time cost during model training mainly results from two parts: (1) the relation-aware information propagation process in the GNN model; (2) the supervision signal detection process.

As for the first part, the time complexity of information propagation over the bipartite interaction graph is $O(L|\mathcal{G}_B|d)$, where L indicates the number of GNN layers, $|\mathcal{G}_B|$ indicates the number of interactions in the user-item bipartite graph, and d denotes the dimension of user embedding. Similarly, the time cost of information propagation over KG is $O(L|\mathcal{G}|d)$, where $|\mathcal{G}|$ denotes the number of triplets in KG.

As for the second part, it depends on pre-trained KGE approaches. Additionally, the time cost of the K-means algorithm also contributes to the time complexity of HIKE. However, we only need to run the pre-trained KGE algorithm one time for each dataset. In other words, once pre-training is done, it will never add any extra time-consuming in model training. Besides, the time cost of K-means is negligible for two main reasons. First, K-means is deployed to run only one time with preset cluster numbers at all levels of KTSs before model training. If the number of cluster numbers at each level remains unchangeable, no extra time is required during model training. Second, the time complexity of K-means, $O(|D|)$ ($|D|$ is the number of entities in KTSs), is pretty small. Taking Beijing dataset as an example, the number of KG triples is more than 366 thousand, while the number of entities in KTSs is less than 400.

Since the pre-trained process will not increase the time costs of each training epoch, the total time complexity of each training epoch is $O(L|\mathcal{G}_B|d + L\mathcal{G}|d)$. When fixing embedding size, the time complexity of HIKE is comparable to it of KGIN [38] and it of KGAT [36].

4 EXPERIMENTS

To evaluate the effectiveness and interpretability of our proposed HIKE, we have conducted extensive experiments to answer the following research questions:

- **RQ1**: How does HIKE perform when compared with state-of-the-art approaches?
- **RQ2**: Since the proposed objective function consists of loss computation in the disentangled space, is it an effective design?
- **RQ3**: How much do important designs (e.g., the number of clusters, the depth of HIKE, etc.) impact model performance?
- **RQ4**: Can HIKE shed light on understanding user interests and provide insights into explicable disentanglement?

4.1 Experiment Settings

4.1.1 Datasets. In this paper, we use two real-world datasets, the Beijing dataset and the Shanghai dataset, which have been collected by Tencent Map [1] with user authorization. Moreover, we filter out items that have interacted

[1]https://map.qq.com/

with less than 10 unique users and users who have engaged with no more than 5 unique items. Detailed statistics of these two datasets are summarized in Table 2. During the experiments, we split the whole dataset into the training set, validation set, and testing set by the ratio of $7:1:2$. We train the model on the training set and evaluate the model performance on the validation set. Once overfitting is detected on the validation set, an early stop will be triggered, and the training process will be ended.

4.1.2 Evaluation Metrics. Metrics used in the evaluation period for personalized recommendation tasks should assess model performance from three aspects: 1) how accurate the candidate sets are on average; 2) how many users can be hit by the personalized candidate sets; 3) from the perspective of ranking, how accurate the candidate sets are. Considering all those three aspects, we choose the following widely adopted metrics [21, 44]: recall, hit ratio (HR), and normalized discounted cumulative gain (NDCG) to access model performance after generating candidate set T_u^n with length n for each user. More specifically, let Test_u denote positive items of user u in the testing set, and $|\cdot|$ indicate the size of the intersection set $\mathrm{T}_u^k \cap \mathrm{Test}_u$, then model performance evaluated by recall is computed as:

$$\mathrm{Recall}@n = \sum_{u \in \mathcal{U}} \frac{|\mathrm{T}_u^n \cap \mathrm{Test}_u|}{|\mathcal{U}|}. \tag{21}$$

Moreover, let indicating function $\mathbb{1}(|\mathrm{T}_u^n \cap \mathrm{Test}_u|)$ equal 1 when the intersection is not empty otherwise 0. Then, model performance HR@n is computed as:

$$\mathrm{HR}@n = \sum_{u \in \mathcal{U}} \frac{\mathbb{1}(|\mathrm{T}_u^n \cap \mathrm{Test}_u|)}{|\mathcal{U}|}. \tag{22}$$

Thus, the larger Recall@n is, the more accurate the recommendation contents are on average; the bigger HR@n is, the more users are hit by the recommender systems.

Besides, to compute NDCG@n, we first compute the ideal discounted cumulative grain (iDCG) for user $u, \forall u \in \mathcal{U}$ as follows,

$$\mathrm{iDCG}@n = \sum_{t=1}^{min(n,|\mathrm{Test}_u|)} \frac{1}{\log_2(t+1)}, \tag{23}$$

which quantify the ideal candidate set when $\mathrm{T}_u = \mathrm{Test}_u$ from the perspective of ranking. Then we evaluate the ranking accuracy of the generated candidate set T_u by computing:

$$\mathrm{DCG}@n = \sum_{t=1}^{n} \mathbb{1}(\mathrm{T}_u^n[t] \in \mathrm{Test}_u) \frac{1}{\log_2(t+1)}, \tag{24}$$

where the indicating function $\mathbb{1}(\mathrm{T}_u^n[t] \in \mathrm{Test}_u)$ equals 1 when the tth item of candidate set T_u also exists in Test_u^n, otherwise 0.

Then, model performance evaluated by NDCG@n is computed as:

$$\mathrm{NDGC@n} = \sum_{u \in \mathcal{U}} \frac{\mathrm{iDCG}@n/\mathrm{DCG}@n}{|\mathcal{U}|}. \tag{25}$$

In this way, the ranking accuracy of candidate sets of all users will be evaluated by metric NDCG. Besides, we fixed the length of the candidate set as $n = 20$ as previous works [10, 36] do in all assessments. To guarantee accurate comparison, we rank all items rather than generating a sampled set for all users [21].

4.1.3 Baselines. We compare the performance of our HIKE with six state-of-the-art approaches. These baselines can be roughly classified into two groups: interaction-based approaches [16, 32, 37] and KG-enhanced approaches [10, 36, 38]. These approaches also include a disentangled model [38], and a very recently published hierarchy-aware model [10].

- **MF-BPR [32]** This model leverages user-item interaction and treats item recommendation as a matrix completion task. It computes pairwise loss to optimize the embedding of both users and items. However, this model only considers first-order connectivity on the user-item bipartite graph.
- **NGCF [37]** This approach implements collaborative filtering on a GNN model. Thus, collaborative signals can be captured from both first-order connectivity and high-order connectivity when the message is propagating on the user-item bipartite graph. Moreover, nonlinear activation functions and transformation matrices are utilized when learning user embedding and item embedding.
- **LGN [16]** This is also a GNN-based model, which simplifies nonlinear activation function and transformation matrices used in traditional GNN-based models. After this simplification, the representative embedding of users and items can be learned via linear transformations, leveraging both first-order connectivity and high-order connectivity.
- **KGAT [36]** This is a KG-enhanced recommendation approach, which recursively propagates embedding on the user-item bipartite graph and KG. An attention mechanism is also employed to discriminate the influence of different neighbors. Compared with KG recommendation approaches which design regularization terms or path extraction rules manually, it exploits high-order paths in an end-to-end fashion elegantly.
- **KGIN [38]** This is a state-of-the-art KG-enhanced recommendation method which exploits user intents behind user-item interactions. User intents are interpreted from the perspective of attentive KG relations, and independence between different intents are guaranteed by regularization term, such as calculating mutual information between different intents.
- **HAKG [10]** This is a GNN-based KG enhanced recommendation model, which learns high-quality user embedding and item embedding in the hyperbolic space. The underlying hierarchical structure in KG can be better learned via its proposed angle constraint and dual item embedding design.

4.1.4 Hyper-parameter Settings. We have implemented HIKE in Pytorch. Following [10, 38], we fix the embedding size of users and items to 64 for HIKE and all baselines. Moreover, to keep the performance comparison fair, we use Adam [20] optimizer and Xiaver [14] initialization for all models. During experiments, the learning rate is tuned among $\{1e^{-4}, 1e^{-3}, 1e^{-2}\}$, and regularization term, such as λ is searched in $\{1e^{-5}, 1e^{-4}, 1e^{-3}\}$ for all methods. Moreover, layer number is tuned among $1, 2, 3$ for GNN-based methods, following [30, 38, 47]. As for batch size, we set 1024 for all approaches. In addition, the number of user intents in KGIN is set to 4 in our experiments, following the best setting published in [38].

4.2 Performance Comparison (RQ1)

We report the performance comparison in Table 3, where the performance of the strongest baseline is presented in an underline format, and the best performance is in bold font with a superscript star. From these experimental results, we can draw the following conclusions:

- **HIKE outperforms all baselines across the two real-world datasets in all cases**. Specifically, HIKE outperforms KGIN, the strongest baseline, by 39.68% and 9.69% w.r.t. Recall@20 in the Beijing dataset and Shanghai dataset, respectively. Meanwhile, HIKE achieves notable improvements in the case of HR@20 by 44.52% on the Beijing dataset. These results have demonstrated the superiority of our HIKE. We ascribe these performance improvements to two major reasons: (1) Benefiting from the supervision signals, the data sparsity problem is alleviated since domain knowledge is highlighted during representation learning. (2) By disentangling user interests and item functions in a hierarchical way, HIKE learns more representative user

Table 3. Overall performance comparison on two datasets.

	Beijing Dataset			Shanghai Dataset		
	Recall@20	NDCG@20	HR@20	Recall@20	NDCG@20	HR@20
MF-BPR	0.01654	0.00216	0.03880	0.06092	0.02249	0.37534
NGCF	0.01478	0.00752	0.03576	0.03491	0.02387	0.22995
LGN	0.01331	0.00156	0.03081	0.05516	0.02054	0.35513
KGAT	0.02083	0.01905	0.04986	0.07139	0.02927	0.42864
HAKG	0.03134	0.01614	0.07396	0.05671	0.04098	0.32546
KGIN	<u>0.04763</u>	<u>0.02464</u>	<u>0.11252</u>	<u>0.09444</u>	<u>0.07203</u>	<u>0.51966</u>
HIKE	**0.06653***	**0.03561***	**0.15473***	**0.10359***	**0.07258***	**0.55474***
%Imp.	39.68%	44.52%	37.51%	9.69%	0.76%	6.75%

embedding and item embedding, which helps it understand user preferences better. Thus, in the prediction period, HIKE generates more satisfying candidate sets for each user.

- **Disentangled representation learning is powerful**. For instance, HIKE and KGIN outperform HAKG and KGAT by more than 30% on average in all cases. We ascribe the performance improvement to the disentangled representation learning in HIKE and KGIN. Since disentanglement can help to distinguish impacts from different factors during decision-making, it can result in better representation learning. Thus, downstream tasks, such as personalized recommendations, can benefit from disentanglement.
- Explicable disentanglement is effective. Specifically, KGIN learns inexplicable disentangled user intents via an attentive module, which combines relations in KG attentively. In contrast, our proposed HIKE, disentangles user interests and item functions according to domain knowledge hierarchically. As reported in Table 3, HIKE outperforms KGIN by 39.68% and 9.69%. Thus, the disentanglement in HIKE is more explicable. Compared with KGIN, HIKE outperforms it significantly in most cases. We attribute this performance improvement to the explicable disentanglement based on domain knowledge. These experimental results have revealed the importance of interpretability in recommender systems.
- **The proposed KTSs-based design is effective**. Compared with HAKG, which utilizes the hyperbolic space to capture the *hierarchical* structure (illustrated in Fig. 1(b)), our proposed method enhances the *hierarchical* structure in KTSs (illustrated in Fig. 1(a)), and outperforms HAKG at all cases significantly. These results suggest the effectiveness of our proposed KTSs-based design. Besides, HAKG doesn't outperform KGIN as reported in [10], and we ascribe this to differences between datasets.
- **KG can help to alleviate the data sparsity problem**. We observe that KG-enhanced recommendation approaches, HIKE, KGIN, HAKG, and KGAT consistently outperform KG-free approaches in all cases on the Beijing dataset by more than 20%, while they outperform KG-free methods in most cases on Shanghai dataset. We attribute these performance improvements to the side information stored in KG.

4.3 Ablation Study (RQ2)

To evaluate the effectiveness of our proposed KTSs-based disentanglement, we have conducted two experiments: (1) Implementing HIKE with disentangled representation learning at all levels via computing disentangled loss $\mathcal{L}^{dis}$, and we termed it as HIKE$_{w/\mathcal{L}^{dis}}$; (2) Implementing HIKE with disentanglement at level k via reserving disentangled loss $\mathcal{L}_k$ only, and we term it as HIKE$_{w/\mathcal{L}^{k}}$, $k = 1, 2, 3$. Relative experimental results are reported in Table 4. In addition, hyper-parameters of HIKE are fixed to the best values reported in Section 4.4. Compared the results from the ablation study with our proposed HIKE, we have drawn the following conclusions:

Table 4. Impact of KTSs-based disentanglement. HIKE$_{w/o\mathcal{L}^{dis}}$ indicates the experiment implemented without disentangled loss at all levels, and HIKE$_{w/\mathcal{L}_k}$ indicate the experiment implemented with loss in the original space and disentangled loss at level k ($k = 1, 2, 3$).

	Beijing Dataset			Shanghai Dataset		
	Recall@20	NDCG@20	HR@20	Recall@20	NDCG@20	HR@20
HIKE$_{w/o\mathcal{L}^{dis}}$	0.04769	0.02437	0.11239	0.09498	0.07227	0.52121
HIKE$_{w/\mathcal{L}_1}$	0.04562	0.02267	0.10804	0.09701	0.07101	0.53375
HIKE$_{w/\mathcal{L}_2}$	0.05218	0.02499	0.12225	0.09707	0.06725	0.53654
HIKE$_{w/\mathcal{L}_3}$	0.04348	0.02105	0.10299	0.09518	0.07093	0.52806
HIKE	**0.06653**	**0.03561**	**0.15473**	**0.10359**	**0.07258**	**0.55474**

- **Our proposed KTSs-based disentanglement is effective, which suggests the rationality of highlighting the importance of KTSs in KG**. To be specific, compared with HIKE$_{w/o\mathcal{L}^{d}is}$, we observe 40% performance improvements on the Beijing dataset and 5% on Shanghai dataset averagely after deployed with the KTSs-based disentangled representation learning. These obvious performance improvements support that utilizing KTSs hierarchically can significantly supplement the feature extraction process based on KG and interaction data.
- **Reserving disentanglement at a specific level only cannot lead to performance improvements in all cases**. Specifically, we find that single disentanglement can improve model performance mostly on the Shanghai dataset but lead to performance degradation slightly in most cases on the Beijing dataset. We ascribe these observations to two major reasons: (i) Although disentangled representation learning process can help to obtain better user embedding and item embedding, we cannot ignore the impacts from different levels. In other words, we think the supervision signals found at different levels supplement each other. Combining all supervision signals of different hierarchies together, the disentangled representation learning can help the original representation learning. Otherwise, leveraging supervision signals at a specific level only might mislead the feature learning, even lead to overfitting in the original space due to incomplete domain knowledge; (ii) We attribute the performance variations to differences between Beijing dataset and Shanghai dataset when deploying disentanglement at a single level only.

4.4 Hyper-parameter Study (RQ3)

In this Section, we conduct experiments to observe the impacts of three important hyper-parameters in model design: learning rate, number of layers, and number of supervision signals. To obtain convincing experimental results, we only tuned the investigated hyper-parameter and fixed other hyper-parameters to the best values. Specifically, the best value of λ in equation (19) is $1e^{-5}$. In the following part, we report the corresponding results when tuning other important hyper-parameters.

4.4.1 ***The impact of learning rate.*** The learning rate is an important hyper-parameter since an inappropriate selection can result in severe performance collapse. Therefore, we tune learning rate among $\{1e^{-2}, 1e^{-3}, 1e^{-4}\}$. In Table 5, we report the performance of HIKE trained with different learning rates. From the reported experimental results, we have observed that $1e^{-4}$ is the best value of the learning rate on both datasets.

4.4.2 ***The impact of model depth.*** The number of layers L is an important hyper-parameter for GNN-based models. In this paper, we search for the best value of L in $\{1, 2, 3\}$. Let HIKE-L indicate a HIKE model with L layer(s). We report performance of HIKE-1, HIKE-2, and HIKE-3 in Table 6. As Table 6 has shown, we observe that HIKE-3 outperforms HIKE-2 and HIKE-1 significantly due to the third-order connectivity being captured additionally. For a similar reason, HIKE-2 outperforms HIKE-1 as well.

Table 5. The impact of learning rate.

	Beijing Dataset			Shanghai Dataset		
	Recall@20	NDCG@20	HR@20	Recall@20	NDCG@20	HR@20
lr= $1e^{-2}$	0.02636	0.01456	0.06320	0.08890	0.06593	0.50703
lr= $1e^{-3}$	0.03302	0.01688	0.07954	0.09716	0.06912	0.53852
lr= $1e^{-4}$	**0.06653**	**0.03561**	**0.15473**	**0.10359**	**0.07258**	**0.55474**

Table 6. The impact of model depth.

	Beijing Dataset			Shanghai Dataset		
	Recall@20	NDCG@20	HR@20	Recall@20	NDCG@20	HR@20
HIKE-1	0.02383	0.01138	0.05668	0.08040	0.05422	0.47763
HIKE-2	0.06303	0.03165	0.14905	0.08822	0.05911	0.50889
HIKE-3	**0.06653**	**0.03561**	**0.15473**	**0.10359**	**0.07258**	**0.55474**

For GNN-based models, increasing the number of layers can help to capture higher-order connectivity on the graph during information propagation and information aggregation periods, which is consistent with our observations. We attribute the performance improvements of stacking layers in HIKE to two main reasons:

- During the information propagation process, higher-order connectivity collaborative signals can be captured on the user-item bipartite graph via layer stacking. In this way, user embedding and item embedding will be more representative; thus, the prediction process will be more representative.
- Information stored in longer KG paths will also be learned on KG. In the Beijing dataset and Shanghai dataset, the length of KG paths often varies from 1 to 3. Due to this characteristic of datasets, increasing the number of layers from 1 to 3 can exactly help our HIKE capture higher-order connectivity on KG. Besides, increasing layer numbers will benefit the information propagation on the user-item bipartite graph since items bridge the bipartite graph and the KG.

4.4.3 ***The impact of cluster numbers.*** As explained in Section 3.2, entity clustering at the same level is utilized, and the output clusters as well as the cluster centers are treated as supervision signals to further implement disentanglement during representation learning. In other words, the number of clusters at level k is exactly the number of supervision signals n_k at level k during disentangled representation learning. Therefore, n_k is an important hyper-parameter, which can affect the quality of disentangled embedding obviously. For the purpose of showing how much model performance is influenced by hyper-parameter $n_k, k = 1, 2, 3$, we conduct experiments and report model performance when n_3 ranges among $\{20, 30, 40, 50\}$ in Fig. 6. As for n_1 and n_2, we fix them to the number of entities at level 1 and level 2, respectively. This setting is reasonable since apparent semantic gaps can be observed among nodes at the first level as well as nodes at the second level. Therefore, these nodes are treated as independent cluster centers when generating disentangled user embedding and disentangled item embedding.

In Fig. 6, we use red line, green line, blue line, and the corresponding color of axes to distinguish model performance at Recall@20, NDCG@20, and HR@20, respectively. On the Beijing dataset and Shanghai dataset, we have observed $n_3 = 30$ is the best value. Moreover, we find that n_3 with a large value as well as a small value can lead to performance degradation on both datasets. We ascribe these experimental results to the following reasons: (1) a large n_3 can result in similar nodes in different clusters, and (2) a small n_3 can lead to dissimilar

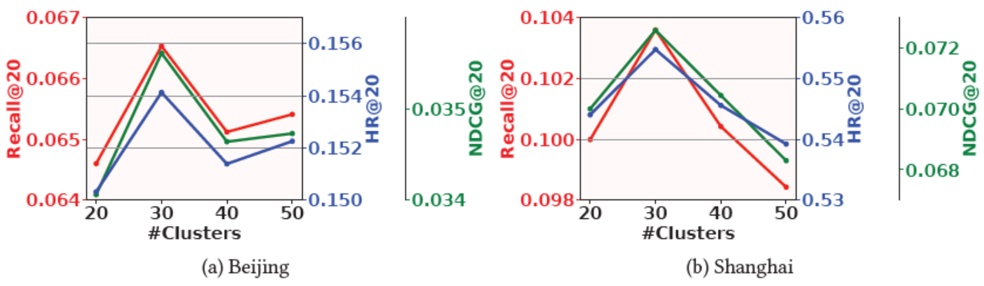

Fig. 6. Impact of the number of supervision signals. In Fig. (a) and Fig. (b), we use the red line, green line, and blue line to indicate model performance at Recall@20, NDCG@20 and HR@20, respectively.

Table 7. Traning data of randomly-selected user u_{20305} in Shanghai Dataset.

Item	Category 1	Category 2	Category 3
Shanghai Disneyland	Leisure Sport	Outdoor Recreation	Amusement Park
Plaza66	Shopping	Integrated Shopping Mall	Shopping: Integrated Shopping Mall
FLASK	Leisure Sport	KTV & Bar &Tea Room & Cinema	Bar
Asakusa 6-chome	Dining	Japanese & Korean Cuisine	Japanese Cuisine
Shangsheng New Institute	Shopping	Commercial Pedestrian Street	Shopping: Commercial Pedestrian Street
Oji cocktail & whisky	Leisure Sport	KTV & Bar &Tea Room & Cinema	Bar
Blackbird	Dining	Western Food	Other Western Food
Shanghai World Expo Museum	Travel & Vacation	Cultural Venues	Museum
THE UNION TRADING CO	Leisure Sport	KTV & Bar &Tea Room & Cinema	Bar

nodes in the same cluster. Thus, the reliability of supervision signals at level 3 will be reduced, which can lead to sub-optimal representation learning and worse recommendation performance ultimately.

4.5 Case Study (RQ4)

As reported in Table 3, our HIKE outperforms all state-of-the-art approaches significantly, which has demonstrated the superiority of HIKE from the perspective of performance improvement. Besides, we will show the powerful interpretability of HIKE using a detailed example in this Section. First, we randomly selected user u_{20305} from the Shanghai dataset. Then we analyzed the similarity between disentangled components of u_{20305} and their corresponding clusters in three levels. Specifically, we compute similarity score $s_{u_{20305}}^{C_{k,j}}$ between user u_{20305} and cluster $C_{k,j}$ at level k as follows,

$$s_{u_{20305},C_{k,j}} = \mathbf{e}_{u_{20305}}^{C_{k,j}} \cdot \mathbf{v}_{k,j}, \quad k = 1, 2, 3, j = 1, \cdots, n_k, \tag{26}$$

where · indicates inner product.

As introduced in Section 4.4.3, nodes' embedding of the first level and nodes' embedding of the second level are used as cluster centers directly due to large semantic gaps. Thus, obtaining similarity scores in the first level and the second level will reflect how closely user behaviors are tied with Cate1 entities and Cate2 entities. During experiments, we observe that the most correlated Cate1 entity and Cate2 entity of u_{20305} are *Shopping* and *Integrated Shopping Mall* . Correspondingly, when looking up the detailed training data of u_{20305} in Table 7, we find that user interests have been captured at the first level and the second level.

Moreover, with respect to the third level, the most relevant cluster is $C_{3,18}$. More specifically, $C_{3,18}$ contains only one Cate3 entity, *Private Kitchens*, which has never appeared in the training data. When we go to check the testing data of u_{20305}, we find that item *Liangsheyeyan*, whose Cate3 is *Private Kitchens* exactly, has been interacted by this user. The results of these analyses above actually have demonstrated the capability of our HIKE in learning explicable disentangled components, as well as predicting users' future interactions. The supervised signals from underlying structures (generated as Section 4.4.3 describes) do benefit the representation learning in disentangled spaces and the original space.

Note that the most frequently engaged Cate1 *Leisure Sport* and $Cate2\ KTV\&Bar\&TeaRoom\&Cinema$ are not identical as the most correlated Cate1 entity and Cate2 entity of user u_{20305}. This difference can also explain why equipped with a single disentangled component can not bring in notable performance improvement (reported in Table 4). Thus, we want to clarify the importance of leveraging the supervision signals at different levels together. In other words, although the obtained disentangled components of a specific user may fail to model his/her interests with extreme accuracy, combining them together can reduce noise during the representation learning process, as well as guarantee gains in downstream tasks.

5 RELATED WORK

5.1 KG-Enhanced Recommendation

KGs could store a lot of real-world facts, common sense, or domain knowledge in the form of triplets, consisting of a head entity, a tail entity, and a relation that connects them. Therefore, in recommendation tasks, item side information (e.g., category, brand) could be used to construct a KG, which helped to alleviate the sparsity problem and increased the explicability of recommender systems. In the last few years, KG-enhanced recommendation approaches have been widely explored and demonstrated their superiority [4, 12, 19, 38, 42]. During the literature review, we found that there were two critical problems when incorporating KG into recommender systems: how to achieve KGE during representation learning and how to distill information from the obtained embedding of KG entities to benefit the downstream tasks.

As for the first problem, the way to implement KGE in existing works could be roughly categorized into two groups: the embedding-based approaches and the path-based approaches. Specifically, embedding-based approaches [3, 18, 22, 43] focused on reserving the similarity of KG triplets in the embedding space. Despite their effectiveness, only first-order connectivity on the KG was leveraged in those translation models. Due to this major drawback, they cannot obtain high-quality embedding. Thus, the downstream tasks could only achieve sub-optimal performance. In contrast, path-based KGE approaches [2, 9, 13, 15] took high-order connectivity into consideration. By manually extracting paths in KG, they could utilize both the first-order connectivity and the high-order connectivity during the KGE process. In this way, the obtained embedding of KG entities was usually more representative and could introduce significant gains into downstream tasks.

In summary, recommendation approaches [1, 4, 35, 45, 48], which incorporated KGE in an embedding-based way, shared the shortcoming of losing information in longer KG paths. Meanwhile, previous works such as [5, 17, 27, 33, 40] that implemented KGE in a path-based way, were effective but rely on the quality of manual path extraction.

Recently, benefiting from graph neural networks [11], we could implement KGE into recommender systems in an end-to-end paradigm [8, 19, 26, 29, 36, 38, 42], considering both kinds of similarities in KG triplets and similarity among KG paths. Specifically, information propagation on KG could be implemented in a path-based manner by stacking layers in a GNN model without manually defining rules for path extraction. Moreover, since items bridged the KG and the user-item bipartite graph, information propagation on the heterogeneous graph could benefit the user feature learning process naturally. Typically, CKAN [42] used a heterogeneous information propagation strategy on the KG and the bipartite graph, and KGAT [36] recursively executed information propagation on these two graphs and learned attentive weights for different neighbors during information aggregation. Moreover, KGIN [38] strengthened the influence of different relations in KG paths and utilized a relational-aware information propagation strategy. However, despite the effectiveness of these works, they paid less attention to leveraging the hierarchical KTSs in KG, which reflected domain knowledge from the coarse-grained level to the fine-grained level. To deal with the underestimation of the impacts of KTSs in existing methods, we have proposed a KTS-based recommendation approach and found that by highlighting the importance of KTSs, more representative user embedding and item embedding could be obtained in the representation learning process.

5.2 Disentangled Representation Learning for Recommendation

Disentangled representation learning originated from the computer vision area. Soon it demonstrated its superiority after being deployed into recommender systems, in which embedding was learned separately due to different factors. This success was not surprising, as decoupling different factors during feature extraction could provide recommender systems with a better understanding of user profiles or item features. Once more representative feature learning was achieved, the accuracy of the matching stage and ranking stage could be improved easily. Thus, more accurate candidate sets were obtained and provided to users. For instance, DisenGCN [24] designed a neighborhood routing strategy, according to which neighborhoods connected due to different factors would be clustered into different groups during training. Moreover, disentangled components corresponding to different factors were learned separately under this routing strategy. Meanwhile, theoretical proof, together with extensive empirical results, have been provided in [24] to evaluate the model performance. Subsequently, [25] and [39] proposed disentangled presentation learning approaches, following the paradigm of learning disentangled components for different factors. More specifically, [25] achieved macro disentanglement between different components as well as micro disentanglement among different dimensions of one component via variational inference; [39] implemented disentanglement via an independence regularization term, which guaranteed user-intents behind disentangled components were not identical. Despite their effectiveness, we found that disentangled components in these methods lacked convincing semantic meanings. In other words, the number of disentangled components was decided by parameter tuning, and the meaning of different components could be either missing or requiring a lot of artificial interpretation. Afterward, [49] proposed a two-factor disentangled approach, in which two explicable components (popularity of items and user preferences) have been decoupled by using a specially-designed sampling strategy. Meanwhile, [38] proposed a KG-enhanced disentangled model, which decoupled different user intents via learning different attentive KG-relation weights. In addition, [41] implemented disentangled embedding of users and items via leveraging different meta paths in KG. Differently, in this paper, we have proposed a disentangled representation learning approach, which utilized the hierarchical KTSs in KG to generate disentangled components. Since KTSs reflected domain knowledge from coarse-grained level to fine-grained level, the obtained disentangled components could reveal user preferences and item functions in different granularity with explicable meanings.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we have highlighted the underestimated significance of KTSs and proposed a KG-enhanced recommendation approach based on a GNN model. Our proposed model learns disentangled user embedding and item embedding hierarchically via information mining from KTSs. Moreover, we designed an objective function, according to which losses will be computed in the original space as well as the disentangled space. Benefiting from the model design and the objective function, our proposed model successfully decouples user interests and item functions in hierarchical levels, giving insights into understanding user preferences. Extensive experiments have been conducted in two real-world datasets. Experimental results have demonstrated the effectiveness and interpretability of our proposed model.

One further improvement of this work is the online test. In the future, we plan to deploy our proposed method into online recommender systems (e.g., product recommendation, POI recommendation, etc.). Thus, the impacts of KTSs will be fully evaluated. Besides, since HIKE obtains supervision signals via pretraining, we plan to deploy the supervision signal detection operation into HIKE in an end-to-end way. In this way, clusters and the corresponding cluster centers will be updated timely, and more accurate supervision signals will be obtained, which can guide the representation learning process to generate better user embedding and item embedding. Thus, domain knowledge reflected by the KTSs can benefit the recommender system in a better way.

ACKNOWLEDGEMENT

This work is supported by the National Key Research and Development Program of China under grant 2022YFB3104702. This work is also supported by National Natural Science Foundation of China under 62272262, U20B2060, U21B2036 and by the Guoqiang Institute, Tsinghua University under grant 2021GQG1005. This work is also supported by the fellowship of China Postdoctoral Science Foundation under 2021TQ0027 and 2022M710006.

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] Qingyao Ai, Yongfeng Zhang, Keping Bi, and W Bruce Croft. 2019. Explainable product search with a dynamic relation embedding model. *ACM Transactions on Information Systems (TOIS)* 38, 1 (2019), 1–29.

[3] 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).

[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] Yibo Chen, Chanle Wu, Ming Xie, and Xiaojun Guo. 2011. Solving the sparsity problem in recommender systems using association retrieval. *J. Comput.* 6, 9 (2011), 1896–1902.

[7] Yankai Chen, Menglin Yang, Yingxue Zhang, Mengchen Zhao, Ziqiao Meng, Jianye Hao, and Irwin King. 2022. Modeling Scale-free Graphs with Hyperbolic Geometry for Knowledge-aware Recommendation. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 94–102.

[8] Zhiyong Cheng, Sai Han, Fan Liu, Lei Zhu, Zan Gao, and Yuxin Peng. 2023. Multi-Behavior Recommendation with Cascading Graph Convolution Networks. In *Proceedings of the Web Conference 2023*. Association for Computing Machinery, 1–10.

[9] Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2016. Chains of reasoning over entities, relations, and text using recurrent neural networks. *arXiv preprint arXiv:1607.01426* (2016).

[10] Yuntao Du, Xinjun Zhu, Lu Chen, Baihua Zheng, and Yunjun Gao. 2022. HAKG: Hierarchy-Aware Knowledge Gated Network for Recommendation. In *Proceedings of the 45th international ACM SIGIR conference on Research and development in Information Retrieval*.

[11] Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, Yingrong Qin, Jinghua Piao, Yuhan Quan, Jianxin Chang, Depeng Jin, Xiangnan He, et al. 2023. A survey of graph neural networks for recommender systems: challenges, methods, and directions. *ACM Transactions on Recommender Systems* 1, 1 (2023), 1–51.

[12] Chen Gao, Yu Zheng, Wenjie Wang, Fuli Feng, Xiangnan He, and Yong Li. 2022. Causal Inference in Recommender Systems: A Survey and Future Directions. *arXiv preprint arXiv:2208.12397* (2022).

[13] Alberto García-Durán, Antoine Bordes, and Nicolas Usunier. 2015. *Composing relationships with translations*. Ph. D. Dissertation. CNRS, Heudiasyc.

[14] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 249–256.

[15] Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. *arXiv preprint arXiv:1506.01094* (2015).

[16] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.

[17] 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.

[18] 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.

[19] Jiarui Jin, Jiarui Qin, Yuchen Fang, Kounianhua Du, Weinan Zhang, Yong Yu, Zheng Zhang, and Alexander J Smola. 2020. An efficient neighborhood-based interaction model for recommendation on heterogeneous graph. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 75–84.

[20] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. (2014).

[21] Walid Krichene and Steffen Rendle. 2022. On sampled metrics for item recommendation. *Commun. ACM* 65, 7 (2022), 75–83.

[22] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*.

[23] Yu Liu, Jingtao Ding, and Yong Li. 2021. Knowledge-driven site selection via urban knowledge graph. *arXiv preprint arXiv:2111.00787* (2021).

[24] Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. 2019. Disentangled graph convolutional networks. In *International Conference on Machine Learning*. PMLR, 4212–4221.

[25] Jianxin Ma, Chang Zhou, Peng Cui, Hongxia Yang, and Wenwu Zhu. 2020. Learning disentangled representations for recommendation. In *Fourteenth ACM Conference on Recommender Systems*. 43–52.

[26] Ting Ma, Longtao Huang, Qianqian Lu, and Songlin Hu. 2022. KR-GCN: Knowledge-aware Reasoning with Graph Convolution Network for Explainable Recommendation. *ACM Transactions on Information Systems (TOIS)* (2022).

[27] 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.

[28] J MacQueen. 1967. Classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Probability*. 281–297.

[29] Shanlei Mu, Yaliang Li, Wayne Xin Zhao, Siqing Li, and Ji-Rong Wen. 2021. Knowledge-Guided Disentangled Representation Learning for Recommender Systems. *ACM Transactions on Information Systems (TOIS)* 40, 1 (2021), 1–26.

[30] Yitong Pang, Lingfei Wu, Qi Shen, Yiming Zhang, Zhihua Wei, Fangli Xu, Ethan Chang, Bo Long, and Jian Pei. 2022. Heterogeneous global graph neural networks for personalized session-based recommendation. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 775–783.

[31] Sung-Jun Park, Dong-Kyu Chae, Hong-Kyun Bae, Sumin Park, and Sang-Wook Kim. 2022. Reinforcement Learning over Sentiment-Augmented Knowledge Graphs towards Accurate and Explainable Recommendation. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 784–793.

[32] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. 456–461.

[33] Zhu Sun, Jie Yang, Jie Zhang, Alessandro Bozzon, Long-Kai Huang, and Chi Xu. 2018. Recurrent knowledge graph embedding for effective recommendation. In *Proceedings of the 12th ACM conference on recommender systems*. 297–305.

[34] Qizhi Wan, Changxuan Wan, Keli Xiao, Rong Hu, and Dexi Liu. 2022. A Multi-Channel Hierarchical Graph Attention Network for Open Event Extraction. *ACM Transactions on Information Systems (TOIS)* (2022).

[35] Chenyang Wang, Min Zhang, Weizhi Ma, Yiqun Liu, and Shaoping Ma. 2020. Make it a chorus: knowledge-and time-aware item modeling for sequential recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 109–118.

[36] 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.

[37] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 165–174.

[38] 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.

[39] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. 2020. Disentangled graph collaborative filtering. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1001–1010.

[40] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable reasoning over knowledge graphs for recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 5329–5336.

[41] Yifan Wang, Suyao Tang, Yuntong Lei, Weiping Song, Sheng Wang, and Ming Zhang. 2020. DisenHAN: Disentangled Heterogeneous Graph Attention Network for Recommendation *(CIKM '20)*. Association for Computing Machinery, New York, NY, USA, 1605–1614. https://doi.org/10.1145/3340531.3411996

[42] 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.

[43] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 28.

[44] Xiwang Yang, Harald Steck, Yang Guo, and Yong Liu. 2012. On top-k recommendation using social networks. In *Proceedings of the sixth ACM conference on Recommender systems*. 67–74.

[45] 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.

[46] Yingying Zhang, Xian Wu, Quan Fang, Shengsheng Qian, and Chengsheng Xu. 2022. Knowledge-enhanced Attributed Multi-Task Learning for Medicine Recommendation. *ACM Transactions on Information Systems (TOIS)* (2022).

[47] Kai Zhao, Yukun Zheng, Tao Zhuang, Xiang Li, and Xiaoyi Zeng. 2022. Joint Learning of E-commerce Search and Recommendation with a Unified Graph Neural Network. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 1461–1469.

[48] Xinxiao Zhao, Zhiyong Cheng, Lei Zhu, Jiecai Zheng, and Xueqing Li. 2021. UGRec: modeling directed and undirected relations for recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 193–202.

[49] Yu Zheng, Chen Gao, Xiang Li, Xiangnan He, Yong Li, and Depeng Jin. 2021. Disentangling User Interest and Conformity for Recommendation with Causal Embedding. In *The world wide web conference*. 2980–2991.

[50] Sheng Zhou, Jiajun Bu, Xin Wang, Jiawei Chen, and Can Wang. 2019. HAHE: Hierarchical attentive heterogeneous information network embedding. *arXiv preprint arXiv:1902.01475* (2019).