KR-GCN: Knowledge-Aware Reasoning with Graph Convolution Network for Explainable Recommendation

TING MA, School of Cyber Security, University of Chinese Academy of Sciences; Institute of Information Engineering, Chinese Academy of Sciences
LONGTAO HUANG, Alibaba Group
QIANQIAN LU, Institute of Information Engineering, Chinese Academy of Sciences
SONGLIN HU, School of Cyber Security, University of Chinese Academy of Sciences; Institute of Information Engineering, Chinese Academy of Sciences

Incorporating knowledge graphs (KGs) into recommender systems to provide explainable recommendation has attracted much attention recently. The multi-hop paths in KGs can provide auxiliary facts for improving recommendation performance as well as explainability. However, existing studies may suffer from two major challenges: error propagation and weak explainability. Considering all paths between every user-item pair might involve irrelevant ones, which leads to error propagation of user preferences. Defining meta-paths might alleviate the error propagation, but the recommendation performance would heavily depend on the pre-defined meta-paths. Some recent methods based on graph convolution network (GCN) achieve better recommendation performance, but fail to provide explainability. To tackle the above problems, we propose a novel method named **K**nowledge-aware **R**easoning with **G**raph **C**onvolution **N**etwork (KR-GCN). Specifically, to alleviate the effect of error propagation, we design a transition-based method to determine the triple-level scores and utilize nucleus sampling to select triples within the paths between every user-item pair adaptively. To improve the recommendation performance and guarantee the diversity of explanations, user-item interactions and knowledge graphs are integrated into a heterogeneous graph, which is performed with the graph convolution network. A path-level self-attention mechanism is adopted to discriminate the contributions of different selected paths and predict the interaction probability, which improves the relevance of the final explanation. Extensive experiments conducted on three real-world datasets show that KR-GCN consistently outperforms several state-of-the-art baselines. And human evaluation proves the superiority of KR-GCN on explainability.

CCS Concepts: • **Information systems → Recommender systems;**

Additional Key Words and Phrases: Explainable recommendation, knowledge graphs, error propagation, graph convolution network

Authors' addresses: T. Ma and S. Hu, School of Cyber Security, University of Chinese Academy of Sciences, Institute of Information Engineering, Chinese Academy of Sciences, Shangdi Street, Beijing, 100093, China; emails: mating@iie.ac.cn, husonglin@iie.ac.cn; L. Huang, Alibaba Group, Donghu Street, Beijing, 100102, China; email: kaiyang.hlt@alibaba-inc.com; Q. Lu, Institute of Information Engineering, Chinese Academy of Sciences, Shangdi Street, Beijing, 100093, China; email: luqianqian@iie.ac.cn.

1046-8188/2023/01-ART4 $15.00
https://doi.org/10.1145/3511019

ACM Reference format:
Ting Ma, Longtao Huang, Qianqian Lu, and Songlin Hu. 2023. KR-GCN: Knowledge-Aware Reasoning with Graph Convolution Network for Explainable Recommendation. *ACM Trans. Inf. Syst.* 41, 1, Article 4 (January 2023), 27 pages.
https://doi.org/10.1145/3511019

1 INTRODUCTION

Recommender systems have played an increasingly important role in online platforms, which aim to alleviate the impact of information explosion and provide personalized recommendation sticking to the interests of users. In recent years, **Knowledge Graphs** (**KGs**), which can provide auxiliary information about users and items in heterogeneous graphs, have been proved to be effective in improving recommendation performance [5, 42, 46, 64].

KGs can organize well-structured external information to connect users and items, which cannot only provide extra facts about items to generate more accurate recommendation, but also expand users' interests to a certain extent. Figure 1 shows an example of the KG enhanced recommendation, where the KG and user-item interactions are integrated. The historical interaction items of users, such as *The Three Musketeers* and *A Tale of Two Cities*, can be linked to entities in the given KG. Then, the attributes of the entities can be utilized to enhance recommendation, such as book titles, writers, genre, and so on. Because historical interaction items and recommended items are connected by entities as well as users, the recommendation enhanced by KGs can explore richer path information than collaborative filtering methods [52, 57]. For example, the book *Barnaby Rudge* can be recommended to *Bob* because they are connected by the entities *Charles Dickens* and *A Tale of Two Cities*, which can hardly be recommended based on users' historical interaction in collaborative filtering methods. Meanwhile, one of the reasoning paths $Bob \xrightarrow{Interact} A\ Tale\ of\ Two\ Cities \xrightarrow{WrittenBy} Charles\ Dickens \xrightarrow{Write} Barnaby\ Rudge$ can be provided as the reason for the explainable recommendation.

A number of recent efforts have attempted to leverage KGs to make the knowledge-aware recommendation. These knowledge-aware recommendation methods can be roughly categorized into two types: path-based methods [19, 62] and embedding-based methods [4, 51]. Although these knowledge-aware methods have achieved promising recommendation performance, challenges still exist, mainly including **error propagation** in the path-based methods and **weak explainability** in the embedding-based methods.

Path-based methods need to identify paths that carry the connectivity information between users and items first, and then feed them into predictive models [10, 19, 38, 47, 50, 64]. However, when modeling the interactions between the given user-item pair, the paths between the user node and the item node are extracted as the propagation paths, and user preferences propagate along these propagation paths in the graph. Considering all paths between the given pair might involve irrelevant ones, which leads to error propagation of the user preferences. Taking Figure 1 as an example, the path $Bob \xrightarrow{Interact} Oliver\ Twist \xrightarrow{Language} English \xrightarrow{Language^{-1}} Barnaby\ Rudge$ has little effect on the recommendation and can be regarded as noise path. Using this path as the reasoning path might introduce noise information into the user preferences, i.e., the error propagation of the user preferences. Defining meta-path patterns [10, 64] might alleviate the problem of error propagation, but the final recommendation performance would heavily depend on the pre-defined meta-paths, which is impossible to reflect the complicated inference relations in reality and there is no way to modify meta-paths during the learning process for optimization. Furthermore, designing proper meta-paths requires a deep understanding of the specific application domain, which is usually labor-intensive and almost impractical to be generalized.

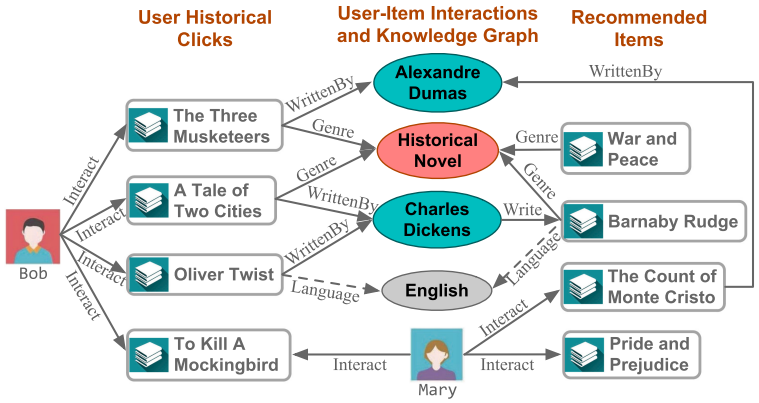


Fig. 1. An example of KG enhanced recommendation, where user-item interactions and KG are integrated. The items are connected with users and entities, so as to explore rich path information and improve the recommendation performance. The dotted line indicates that the path has little effect on the recommendation and may even introduce noise, leading to error propagation.

Embedding-based methods tend to distill user preferences by mapping the embeddings of users and items into a preference score. Specifically, more and more recent works start to utilize **Graph Convolution Network (GCN)** [23] for computing the embeddings of users and items via propagation and aggregation of their neighbors in the heterogeneous graphs [45, 46]. Such GCN-based recommendation methods have proved to be successful in modeling the high-order relations in KGs to provide better recommendations. And the state-of-the-art performance on public datasets is mostly achieved by GCN-based methods. However, these GCN-based methods also have some disadvantages. They compute the preference score only from the embeddings of users and items, lacking much key information on the paths for multi-hop reasoning. Moreover, the explainable recommendation can improve the acceptance and conversion rates of the recommended items. GCN-based methods cannot output multi-hop paths as reasons for the recommendation, failing to provide explainability.

To tackle the aforementioned problems, we propose a novel recommendation model named ***Knowledge-aware Reasoning with Graph Convolution Network*** **(KR-GCN)**, which aims to study how accurate recommendation and trustworthy explainability can be achieved at the same time. Specifically, to cope with the problem of error propagation, we design a transition-based method to determine the triple-level scores and utilize the nucleus sampling strategy [18] to select triplets within the reasoning paths between every user-item pair adaptively. This strategy not only alleviates the problem of error propagation, but also releases the domain knowledge dependency on the pre-defined meta-paths. To improve the recommendation performance and guarantee the diversity of explanations, we integrate user-item interactions in recommendation and the KG (i.e., user collaboration information and auxiliary knowledge about items) into a heterogeneous graph, where the high-order relations in the heterogeneous graph are performed by the GCN. A path-level self-attention mechanism is applied in our framework to discriminate the different contributions of selected paths. The path with the highest weight is provided as the explanation for recommendation, so as to improve the relevance of the final explanation. For example, providing the explanation "The user who has bought the *item A* has also bought *this item*" is more persuasive than the explanation "The user who has seen the *item A* has also seen *this item*" for recommendation. The interaction probability between the given user and the candidate item is predicted via aggregating these selected path representations with various attention scores. To evaluate the recommendation performance of the proposed model KR-GCN, we conduct a serial of experiments

on three public benchmarks. The results indicate that our proposed KR-GCN outperforms state-of-the-art methods like RippleNet [42], KGAT [46], and JNSKR [5]. And human evaluation proves the superiority of the proposed model on explainability.

The main contributions of this article can be summarized as follows:

— We propose a novel knowledge-aware reasoning model named KR-GCN for the explainable recommendation. A transition-based method is designed to score triplets, and nucleus sampling is utilized to select triplets within the paths between every user-item pair adaptively, which can reduce the impact of error propagation and does not rely on domain knowledge.
— To improve the recommendation performance and provide trustworthy explainability, GCN is performed to learning node representations of the heterogeneous graph, including user-item interactions and KG, which can guarantee the diversity of explanations. Then a path-level self-attention mechanism is applied to discriminate the different contributions of selected paths, which improves the relevance of the final explanation.
— Extensive experiments on three real-world datasets about books, business, and music demonstrate that the proposed KR-GCN achieves improvements over state-of-the-art baselines. And human evaluation proves that KR-GCN can provide trustworthy explainability compared with baselines.

The rest of this article is organized as follows. In Section 2, we provide a comprehensive overview of the related work, including KG-aware recommendation, explainable recommendation, and **graph neural network** (**GNN**). In Section 3, we formulate the knowledge-aware recommendation problem and describe the details of our proposed model KR-GCN. The experimental results and analyses of the recommendation performance and explainability are presented in Section 4. Finally, we conclude our work and plan future research directions in Section 5.

2 RELATED WORK

In this section, we review the related work of the proposed KR-GCN, mainly including the KG-aware recommendation, the explainable recommendation, and the GNN.

2.1 KG-Aware Recommendation

The related studies of KG-aware recommendation can be grouped into path-based and embedding-based methods. Path-based recommendation methods [10, 19, 20, 29, 36, 47, 50] usually utilize multi-hop paths in KGs to improve recommendation performance. Moreover, these multi-hop paths can also be used as the propagation paths of user preferences, which can represent the propagation of user preferences more intuitively. MCRec [19] and MEIRec [10] explicitly encode meta-paths as interactions between the users and the items, where the meta-paths are pre-defined according to the recommendation datasets. KPRN [47] extracts multi-hop paths from KGs for recommendation and generates path representations that contain the information of entities, entity-type, and relations. PGPR [50] finds the multi-hop paths between users and items via reinforcement learning and provides recommendation results along the reasoning paths. RuleRec [29] proposes a joint learning framework to generate the rule-guided neural recommendation, where the joint learning framework consists of a rule learning module and a recommendation module. The work [64] devises a meta-heuristic based demonstration extractor and perform path finding by leveraging these path demonstrations.

Embedding-based recommendation methods [1, 4, 43, 51, 54, 58] usually utilize **knowledge graph embedding** (**KGE**) methods, such as TransE [3], TransH [48], DistMult [53], and ComplEx [40], to learn the entity embeddings in KGs for the corresponding items and enhance the representations for user-item interactions. CKE [58] adopts the KGE method TransR [27] to extract KG

information and then combines the KG representations with textual and visual representations to generate multi-modal representations for items. DKN [43] learns the entities contained in the news via KGE methods and incorporates these entity embeddings in the KGs into news recommendation to capture the connections among different news, so as to perform click-through rate prediction. KTUP [4] joins the tasks of item recommendation and KG completion to enhance the item embeddings and user preferences, where the user preferences are induced by the user-item interactions and the relations in KG. Some models [32, 39, 42, 44–46, 55, 63] incorporate path features into embedding-based methods to perform recommendation. RippleNet [42] models the propagation of user's preference from historical interests along paths in the KG to predict the user clicking probability. KGCN [46] utilizes GCN to embed the entities in the KG, and then use these entities to capture users' interests. KGAT [45] models the high-order connections between user-item pairs with embeddings refined by recursive propagation.

However, the path-based recommendation methods usually ignore the graph structure information when capturing multi-hop paths, and most methods that utilize meta-paths to model user preferences are difficult to reflect the complicated inference relations and inefficient to optimize in reality. Furthermore, defining meta-paths requires domain knowledge and is usually labor-intensive. Although the embedding-based recommendation methods are flexible and efficient, they cannot output multi-hop paths as recommendation reasons, failing to provide explainability.

2.2 Explainable Recommendation

Explainable recommender systems [1, 6, 8, 9, 12, 13, 20, 26, 28, 29, 47] have played an increasingly important role in recommendation task, or even in machine learning. This is mainly because the explainable recommendation can greatly improve the effectiveness of recommendation (to help users make decisions quickly) and the persuasiveness of recommendation (to improve the possibility of users accepting or buying recommended items, i.e., acceptance and conversion rates). The explainable recommendation can provide a better interactive experience for users and bring considerable benefits to application manufacturers. KPRN [47] extracts multi-hop reasoning paths from KGs to infer user preferences. These multi-hop reasoning paths are assigned different weights and used as explanations for the recommendation results. EIUM [20] captures the semantic paths between specific user-item pair in KGs. Then these multi-hop paths are encoded and weighted, so as to provide the path-wise explanations for the sequential recommendation. RuleRec [29] is a rule-guided recommendation method that utilizes the KG as an information source. RuleRec induces rules via a rule learning module and then explores item associations between item pairs based on these deduced rules, where the associations between item pairs are used as explanations. To improve the fairness of explainable recommendation, [12] proposes a fairness-aware algorithm to alleviate the algorithmic bias and perform fair explainable diversity. DEAML [13] utilizes an attentive multi-view learning framework to combine predictions with different weights and proposes a dynamic programming algorithm to generate explanations for the personalized recommendation. To perform the explainable recommendation, TMER [6] captures meta-path patterns between items to explore the dynamic evolutions of user-item pairs based on temporal KGs. DESR [26] uses the **Gaussian Mixture Model** (**GMM**) and the capsule network to obtain users' long-term preferences and short-term demands, and combines these two via the serendipity vector. Then DESR devises a back-routing scheme to provide explanations for the recommendation. The work [8] uses the multi-task learning framework to learn the recommendation task and the explanation task, and designs a hierarchical co-attention selector to model the interaction between two tasks. The work [64] proposes a demonstration-based explainable recommendation method, which extracts reasoning paths by using several path demonstrations. AnchorKG [28] performs news recommendation

by generating anchor KG for each article, and provides explanations in the form of multi-hop relational reasoning paths.

Existing explainable recommendation methods usually provide the explainability by extracting paths from the KGs and then encoding path information via some sequence algorithms, such as RNN and LSTM. These methods can provide explanations for the recommendation, but the graph structure information is not fully utilized. The graph structure information can reflect the influence of nodes by their neighbors, so using the graph structure information can improve user-item interaction modeling to some extent.

2.3 Graph Neural Network

In recent years, the GNN has shown great potential in representation learning, and has achieved considerable performance in many natural language processing tasks, such as text classification [34, 61] and recommendation system [7, 31, 60], and so on. GNN aims to model the graph structure information and node representation information. The nodes in the graph are represented as low-dimensional vectors, and then the learned node representations are used for the downstream tasks by employing neural network models. Several typical GNNs include GCN [23], **Graph Attention Networks** (**GAT**) [41], **Graph Auto-Encoders** (**GAE**) [22], Graph Generative Networks [24], and so on. In the recommendation task, GNN is typically employed to learn the node representations for users and items. GraphRec [11] utilizes the GNN framework to embed the nodes and interactions in the social graph and the user-item graph for the social recommendation. Three aggregators are devised to perform node aggregation in two heterogeneous graphs. V2HT [25] introduces a multi-view interactive feature recommendation framework based on graph information propagation, which aims to alleviate the problem of data sparsity and long-tail distributions. DGRec [37] employs dynamic graph attention neural network to perform social recommendation, where the dynamic user behaviors and context-dependent social influence are utilized to infer users' interests. KGCN [45], KGCN-LS [44], and KGAT [46] automatically discover the high-order connections in the KG and update node representations by aggregating the neighborhood information. DANSER [49] uses two dual GAT for the social recommendation. The GAT in DANSER are composed of a user-specific dual GAT and a dynamic and context-aware dual GAT. STAR-GCN [59] learns the node representations in the graph via a multi-link graph convolutional encoder. To enhance the representations and alleviate the cold start problem, STAR-GCN masks a few nodes and reconstructs these nodes via their neighborhood information. IntentGC [63] adopts GCN to capture users' preferences and heterogeneous relationships, and devises a vector-wise convolution function to perform faster recommendation.

Although the GCN-based methods have proved to be successful in modeling the high-order relations in KGs to provide better recommendation. And the state-of-the-art performance on public datasets is mostly achieved by GCN-based methods. However, these GCN-based methods also have some disadvantages. They compute the preference score only from the embeddings of users and items, lacking much key information on the paths for multi-hop reasoning and failing to provide explainability.

Different from previous KG-aware and explainable recommendation methods, KR-GCN not only makes full use of the graph structure information, but also adaptively captures multi-hop paths in the graph to solve the problem of error propagation without defining meta-path patterns. The heterogeneous graph and the multi-hop paths are encoded via GCN and LSTM, respectively. Since the heterogeneous graph that we construct includes the user-item interactions and the KG, KR-GCN can guarantee the diversity of explanations compared with the methods that rely solely on either the KG or the user-item interactions. Furthermore, the relevance of the explanations is improved via the path-level self-attention mechanism in KR-GCN.

3 METHODOLOGY

In this section, we present the proposed recommendation model KR-GCN. Firstly, we formulate the knowledge-aware recommendation problem. Then, we elaborate on the core components of the proposed model KR-GCN, including Graph Encoding module, Path Extraction and Selection module, Path Encoding module, and Preference Prediction module, as illustrated in Figure 2. Lastly, we describe the training and optimization of KR-GCN. Specifically, we define the training objective function of our model.

3.1 Problem Formulation

The goal of our work is to learn a prediction function for user preferences that can predict the interaction probability between the given user and the target item, i.e., whether the user will interact with the item that he has not interacted with before. Given a user set $\mathcal{U}$ and an item set $\mathcal{V}$, for the user $u \in \mathcal{U}$, the items that u interacts with are denoted as $V_u = \{v_1, v_2, \ldots, v_M\} \subset \mathcal{V}$, where M denotes the number of u's historical interaction items.

The KG $\mathcal{G}(\mathcal{E}, \mathcal{R})$ is introduced as the complementary information and organized as structured triples, where $\mathcal{E}$ and $\mathcal{R}$ are the entity set and relation set. In the KG enhanced recommendation, the item $v \in \mathcal{V}$ is linked to the corresponding entity $e \in \mathcal{E}$ in the given KG $\mathcal{G}$ (Noting that in the news recommendation scenario, an item corresponds to many entities). Then, the knowledge can be incorporated into the recommendation by introducing the triplets that contain the entity e. Taking Figure 1 as an example, the item *Oliver Twist* can be linked to the entity *OliverTwist*, which relates to other entities in the form of triplets in $\mathcal{G}$, such as the entity *Charles Dickens*. Given the historical interaction items V_u of u, the target item $v \notin V_u$, as well as the KG $\mathcal{G}$, the prediction function for user preferences can be defined as

$$\hat{y}_{uv} = f_\Omega(u, v | V_u, G), \tag{1}$$

where $\hat{y}_{uv}$ denotes the interaction probability between the user u and the target item v, and f is our recommendation model with the parameter set Ω.

3.2 KR-GCN

The proposed recommendation model KR-GCN consists of four modules: Graph Encoding, Path Extraction and Selection, Path Encoding, and Preference Prediction. The architecture of KR-GCN is illustrated in Figure 2. The Graph Encoding module is designed to learn the representations of nodes in the heterogeneous graph. The Path Extraction and Selection module is devised to extract paths between users and items from the heterogeneous graph and select higher-quality reasoning paths. The Path Encoding module is used to learn the representations of the selection reasoning paths. The Preference Prediction module predicts users' preferences according to the reasoning paths.

3.2.1 Graph Encoding. To accommodate embeddings for users, items, and entities, KR-GCN encodes the heterogeneous graph including user-item interactions and KG by utilizing the graph representation model GCN. The embeddings of nodes are computed via performing graph convolution iteratively and aggregating local network neighborhood information. Then, the structure information of the heterogeneous graph can be modeled and the high-order connectivities in the graph can be captured. Moreover, because GCN updates each node embeddings by utilizing adjacent node embeddings, the problem of node ambiguity can be solved well. For example, the nodes *Dumas Jr.* and *Dumas Sr.* can be distinguished according to their neighborhood nodes *The Lady of the Camellias* and *The Three Musketeers* in the graph.

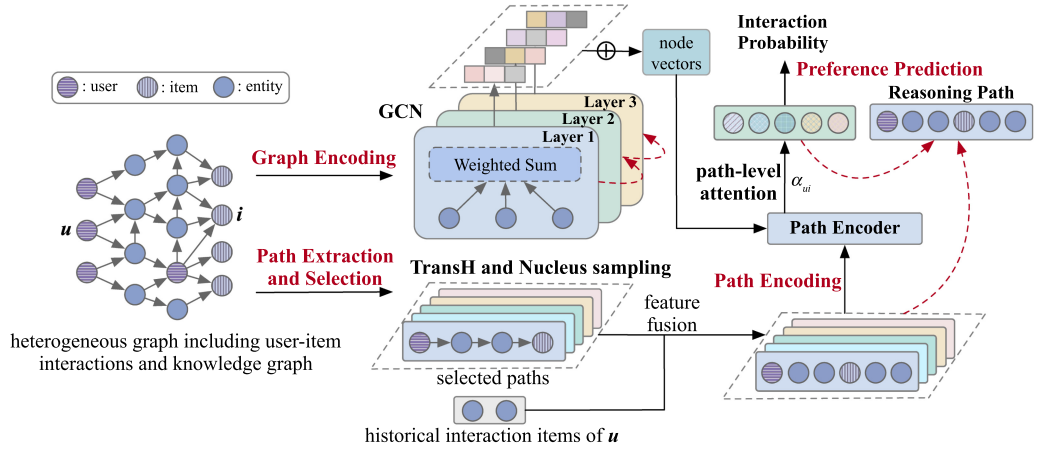


Fig. 2. An illustration of KR-GCN model architecture. Four modules are included: Graph Encoding, Path Extraction and Selection, Path Encoding, and Preference Prediction.

In KR-GCN, we adopt the weighted sum aggregator to capture the features of each given node and their neighborhood nodes, where the neighborhood nodes are aggregated via mean function. The sum aggregator combines these two representations with a non-linear activation function σ. Specifically, for a given node i (i.e., a user, an item, or an entity), the embeddings can be initialized randomly or using original node features pre-trained with external knowledge at the 0th layer. At higher layers, the node embeddings are computed via graph convolution operation. The operation on the node i at the $(l+1)$th layer can be abstracted as

$$e_i^{(l+1)} = \sigma\left(W_{self}^{(l)} e_i^{(l)} + \sum_{j \in N_i} \frac{1}{|N_i|} W^{(l)} e_j^{(l)}\right), \tag{2}$$

where $e_i^{(l+1)}$ and $e_i^{(l)}$ are the embeddings of the node i at the $(l+1)$th layer and lth layer, respectively. N_i is the set of i's neighborhood nodes and $e_j^{(l)}$ is the jth neighborhood node of i at the lth layer. $W_{self}^{(l)}$ and $W^{(l)}$ are the transformation weight matrices of i and i's neighborhood node, respectively. After L layers convolution operations, there are L representations for the given node i. Then, the weighted sum operation is applied for the embeddings that calculated at each layer, i.e., from $e_i^{(1)}$ to $e_i^{(L)}$, so as to encode the given node i. Let e_i denote the final representation of the node i, and the weighted sum calculation of i's representation is defined as

$$e_i = \sum_{l=0}^{L} \alpha_l e_i^{(l)}, \tag{3}$$

where α_l denotes the weight of the lth layer, i.e., the importance of the lth layer to the final target node representation. Combining the embeddings that are calculated at each layer can help capture different semantic information and make the representation more comprehensive [16].

3.2.2 Path Extraction and Selection. The latent relations among nodes are the high-order connectivities in the graph, which can be extracted and formed as multi-hop paths explicitly between node pairs, such as user-item pairs. Because the high-order connectivities between the given users and the candidate items can reflect the potential interests of users, we explicitly extract multi-hop paths between each user-item pair over the heterogeneous graph to obtain the representations of

the user's potential interests. Reasoning on paths suffers from the problem of error propagation, because considering all paths might involve the irrelevant ones. Defining meta-paths might alleviate the problem of error propagation, but designing proper meta-paths requires a deep understanding of domain-specific knowledge, which is labor-intensive and almost impractical to be generalized. To cope with the error propagation and the knowledge dependence issues, we prune irrelevant paths between each user-item pair. For each user-item pair (u, v), where $u \in \mathcal{U}$ and $v \in \mathcal{V}$, we can efficiently find reasoning paths between u and v over the graph and form the selected paths as a path set S_{uv}. Since the number of paths between the user-item pairs grows exponentially with path hops, we extract multi-hop paths with the limitation that hops in every single path are less than l. Following the setting in the work of [47], we set $l = 3$ in the experiment to gather three-hop paths. The jth path in the path set S_{uv} can be described as

$$S_{uv}[j] = [i_1, r_1, i_2, r_2, \ldots, i_n], \tag{4}$$

where i is the single node (i.e., a user, an item, or an entity) and r is the relation that connects two nodes within the path $S_{uv}[j]$. $i_1 = u$, $i_n = v$, and n is the number of nodes within $S_{uv}[j]$.

Considering that iterating all paths for each user-item pair is inefficient in real-world large-scale KGs, we use the heuristic path search algorithm for path extraction and selection. Specifically, we design a transition-based method to determine the triple-level scores and utilize nucleus sampling to adaptively select triples within the paths between every user-item pair. This strategy captures the transition features within a path and does not rely on any meta-path patterns. We use Δ_{k-1} and Δ_k to denote the selected node sets in the $k-1$th hop and the kth hop of path search. For the node i_{k-1} in the node set Δ_{k-1}, we search its neighbors in the graph as next hop nodes of the node i_{k-1}. For the neighbor node i_k, the score of the corresponding triplet $(i_{k-1}, r_{k-1}, i_k) \in T_{k-1,k}$ is calculated via the KGE method to measure the quality, where $T_{k-1,k}$ is the triple set between the $k-1$th and the kth hops.

In knowledge-aware recommendation, the recommendation data includes user-item interactions and KG, and there are associations and constraints within the triple in the KG. In this article, the scores for all triplets are calculated via the KGE method TransH [48]. The KGE methods, such as TransE and TransH, can be used to measure the quality of the paths by calculating the scores of the triplets within the paths, so as to prune irrelevant paths from noisy graphs [56]. TransH is a transition-based KGE method, which associates each relation with a relation-specific hyperplane and projects entity vectors on that hyperplane. The main reasons we choose TransH include: (i) Many transition-based KGE methods, such as TransH, aim to model associations and constraints within triples, which are simple but efficient; (ii). TransH can deal with one-to-many, many-to-one, and many-to-many relation patterns in the graph. Considering that these relation patterns widely exist in user-item interactions and KG, for example, a user clicks on multiple movies, or a singer sings multiple songs, TransH is a proper choice to calculate the scores of triples in the path extraction and selection module. In the experiment, TransH is trained in advance. For simplicity, we use (h, r, t) to denote the representations of target triplet, and the score calculation of the triplet is defined as

$$f(h, r, t) = -d(h, r, t), \tag{5}$$

$$d(h, r, t) = ||h_\perp + r - t_\perp||_2^2, \tag{6}$$

where $f(h, r, t)$ is the score function, and $d(h, r, t)$ is the distance function. The vectors $h_\perp$ and $t_\perp$ are obtained by projecting h and t on the hyperplane of the relation r:

$$h_\perp = h - w_r^T h w_r, \tag{7}$$

$$t_\perp = t - w_r^T t w_r, \tag{8}$$

where w_r^T and w_r are normal vector of r.

For the purpose of comparison, we also choose TransE [3] and DistMult [53], two typical KGE methods, as triplet scoring functions. Different from TransH, TransE assumes that the relation r is a translation vector connecting the embedded head entity h and tail entity t. The distance function of (h, r, t) in TransE is defined as

$$d(h, r, t) = ||h + r - t||_2^2. \tag{9}$$

DistMult represents each relation as a diagonal matrix and projects the head entity vector to the tail entity vector via this relation matrix. The distance function of (h, r, t) in DistMult is defined as

$$d_r(h, t) = -h^T M_r t, \tag{10}$$

where M_r denotes the projection matrix associated with the relation r, and h^T is the transposed vector of the vector h. In DistMult, the larger the $h^T M_r t$ is, the greater the plausibility of the triplet, which is different from TransE and TransH. Therefore, we add a negative sign before the distance function of DistMult to be consistent with the methods TransE and TransH.

After calculating the score of the triplet (i_{k-1}, r_{k-1}, i_k), we utilize nucleus sampling [18] to adaptively select triples within the paths between every user-item pair. TransH and the nucleus sampling are used to perform path ranking and selection, and then solve the problem of error propagation, i.e., filtering low-quality paths. The KGE methods (such as TransE and TransH) can be used to measure the quality of the paths by calculating the scores of the triplets within the paths [56]. Specifically, the triplets are ranked according to their scores calculated in formulas (5) and (6), so as to prune irrelevant triplets as well as paths between each user-item pair. Nucleus sampling aims to sample the top-p portion of the candidate probability distribution adaptively. Our goal is to make low-quality paths score lower through formulas (5) (6) and filter them. For example, the path *Oliver Twist* $\xrightarrow{Language}$ *English* $\xrightarrow{Language^{-1}}$ *Barnaby Rudge* in Figure 1 is a low-quality path. The formulas (5) and (6) are mainly used to model associations and constraints within triples. The higher the semantic association within the triple (i.e., confidence), the higher the score of the triple. Then, the probability of the path being selected is greater; that is, the triples with higher scores contribute more to the path selection. Therefore, TransH and the nuclear sampling strategy mainly filter out the paths with low confidence.

Instead of setting a fixed number of samples and sampling from the triple sets, such as top-k sampling [33], the number of samples for nucleus sampling is determined by the sum of probability values. At each hop, the triples are selected from the smallest possible triples set whose cumulative probability exceeds a threshold, where the cumulative probability is calculated via summing the triples' probability scores. In this way, the number of sampling triples in the triple set can be dynamically increased or decreased based on the probability distribution. In order to perform nuclear sampling, the triple scores are normalized to compute the probabilities of the triples. The probability score of the triple (i_{k-1}, r_{k-1}, i_k) is obtained via softmax function:

$$P(i_{k-1}, r_{k-1}, i_k) = \frac{\exp(f(e_i^{k-1}, e_r^{k-1}, e_i^k))}{\sum_{(i'_{k-1}, r'_{k-1}, i'_k) \in T_{k-1,k}} \exp(f(e_{i'}^{k-1}, e_{r'}^{k-1}, e_{i'}^k))}, \tag{11}$$

where e_i^{k-1}, e_r^{k-1}, e_i^k are the vectors of i_{k-1}, r_k, and i_k, $f(e_i^{k-1}, e_r^{k-1}, e_i^k)$ is the score of the triple (i_{k-1}, r_{k-1}, i_k) calculated by TransH, $(i'_{k-1}, r'_{k-1}, i'_k)$ is the triple in the set $T_{k-1,k}$. Given the probability distribution of all triples between the $k-1$th hop and kth hop, the selected triples $topp(T_{k-1,k}) \subset T_{k-1,k}$ are defined as the smallest set that satisfies the following conditions:

$$\sum_{(i_{k-1}, r_{k-1}, i_k) \in topp(T_{k-1,k})} P(i_{k-1}, r_{k-1}, i_k) >= p, \tag{12}$$

where p is the probability threshold. Then, the triples in $topp(T_{k-1,k})$ are selected as the reasoning triples within the reasoning paths. At each hop, the triples are selected in the same way as described above. Finally, the reasoning path set S_{uv} can be formed to reflect the user u's potential interests, so as to alleviate the effect of error propagation.

3.2.3 Path Encoding. Inspired by [65], each user's historical interaction items are encoded and concatenated with corresponding embedded paths to enhance the representations of the multi-hop paths, so as to reflect the user's interests. In KR-GCN, the historical interaction set V_u serves as an additional input. Although S_{uv} already contains the path information between u and v, these paths are mainly for the item v and cannot reflect other interests of the user u. For example, if u clicks on a comedy and a documentary, u' interest in the documentary is not considered when exploring the correlation between u and comedy. To explore more interests of users, the mutual effects between selected paths and the user's historical interactions are captured by incorporating the user's historical interactions into selected paths. Moreover, the mutual effects will be weakened if their embeddings are combined after encoding by path encoder separately, because the later the combination is, the weaker the semantic interaction. Considering that there is no timestamp that corresponds to user-item interactions in the datasets, we combine the historical interaction items with selected paths without exploiting the chronological order of historical interaction items. To capture the mutual effects between selected paths and the user's historical interactions, we design a method of sequence combination, namely sequence concatenation. The combination of the path sequence $S_{uv}[j] \in S_{uv}$ and the historical interaction set V_u can be described as

$$T_{uv}[j] = S_{uv}[j] \oplus V_u, \tag{13}$$

where $\oplus$ is the sequence concatenation operation, and $T_{uv}[j]$ is the concatenated sequence.

KR-GCN utilizes the stack of the **long short-term memory** (**LSTM**) and the attention network to encode the selected reasoning paths based on the embeddings learned via graph convolutional network. The LSTM path encoder is utilized for encoding the heterogeneous graph with respect to the multi-hop reasoning paths between the user-item pairs. This module takes the outputs of the graph encoding module and the path extraction and selection module as input. The graph encoding module provides node representations, and the path extraction and selection module provides path information for the path encoding module. Because there are multi-hop relational information and sequential dependencies between different nodes in the paths, this module aims at capturing the multi-hop relational information and encode the sequential dependencies within every single path. The path sequence $S_{uv}[j]$ is initially embedded as $[e_1, e_2, \ldots, e_n]$, which is computed via convolution and aggregation operations in graph convolutional networks. For the path sequence $S_{uv}[j]$, the hidden state at the time slice t in LSTMs can be described as follows:

$$h_t = LSTM(h_{t-1}, e_i), \tag{14}$$

where h_t is the representation of the node i_t. Then attention mechanism is performed to choose important features within the path $S_{uv}[j]$, which returns a vector to represent the single path sequence $S_{uv}[j]$.

$$\alpha_{h_t} = \frac{\exp(ReLU(wh_t + b))}{\sum_{t'=1}^{n} \exp(ReLU(wh_{t'} + b'))}, \tag{15}$$

$$P_{uv}[j] = \sum_{t=1}^{n} \alpha_{h_t} \cdot h_t, \tag{16}$$

where $p_{uv}[j]$ is the learned representation of the selected path $S_{uv}[j]$ between the user u and the item v, and α_{h_t} denotes the importance of the node i_t to the path $S_{uv}[j]$. Then the multi-hop

reasoning paths (or latent relationships) S_{uv} between the user u and the item v are represented by a set of vectors p_{uv}. These path representations can reflect the propagation of u's potential interests.

3.2.4 Preference Prediction. In recommendation, different paths usually make varying contributions to predict user preferences. To discriminate the different contributions of different paths between each user-item pair on reasoning, a path-level self-attention mechanism is applied in KR-GCN. With path-level self-attention, the path-specific weight over each path can be learned. Then, the different importance of different paths can be obtained. And after that, all selected multi-hop paths with different weights are aggregated to represent the user's preferences. The path-level self-attention mechanism is implemented as follows:

$$P_{uv} = softmax\left(\frac{QK^T}{\sqrt{d}}\right)V, \tag{17}$$

$$p_{uv} = maxpool(P_{uv}), \tag{18}$$

where p_{uv} is the representation of the path set S_{uv} through the self-attention mechanism and max-pool operation. $softmax(\frac{QK^T}{\sqrt{d}})$ aims to normalize representations into the probability distribution, which returns the importance of each path on reasoning. $maxpool(P_{uv})$ is used to learn vector representation for the path set S_{uv} between the user u and the candidate item v. Q, K, and V denote query, key, and value representations, and are calculated as follows:

$$Q = W_Q E, \tag{19}$$

$$K = W_K E, \tag{20}$$

$$V = W_V E, \tag{21}$$

where W_Q, W_K, and W_V denote the weight matrices, E is the input matrix of the path set S_{uv}. Finally, we conduct the **Multi-Layer Perceptron** (**MLP**) layers and an activation function on the output P_{uv} to compute the preference prediction score between the user u and the item v, i.e., the probability of the user u interacting with the candidate item v.

$$\hat{y}_{uv} = \sigma(MLP(p_{uv})), \tag{22}$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$ is the sigmoid function, and $MLP(\cdot)$ is the MLP layers with one output. The final prediction score $\hat{y}_{uv}$ is the interaction probability between the user u and the item v, i.e., the score of user preference prediction.

After computing the interaction probability of the given user-item pair, the reasoning path with the highest weight is outputted as the explanation for the recommendation.

3.3 Training and Optimization

In the recommendation models, observed interactions are usually set as positive instances, and unobserved interactions are set as negative instances. To train our recommendation model, we employ the **Bayesian Personalized Ranking** (**BPR**) loss [35] in KR-GCN. BPR loss function is a pairwise loss function. Specifically, BPR loss is devised to predict a positive instance higher than a negative one, where the negative instance is randomly sampled from all items (removing the items that the user has interacted with). The following objective function is minimized to train our model:

$$\mathcal{L}_{BPR} = \sum_{(u,v)\in\Delta} \sum_{(u,v')\in\Delta'} -ln\sigma(\hat{y}_{uv} - \hat{y}_{uv'}) + \lambda||\Omega||_2^2, \tag{23}$$

where $\sigma(\cdot)$ is the sigmoid function, Δ and Δ' are the sets that contain positive and negative user-item interactions, respectively. The parameter set Ω mainly contains learned embeddings in the model as well as hyper-parameters, such as the weight matrices and bias. $\lambda||\Omega||_2^2$ is conducted to prevent overfitting of the model, where λ is the regularization parameter. The optimization of the above parameters is alternatively performed using Adam optimizer [21] with mini-batch.

4 EXPERIMENTS

In this section, we evaluate the effectiveness of the proposed KR-GCN on three real-world datasets about books, music, and business. In the following subsections, we first introduce the experimental setup briefly, mainly including the description of the datasets, baselines, evaluation metrics, and implementation details. Then, the performance comparison and analysis will be shown, in which we compare the proposed KR-GCN with several baselines, discuss the effectiveness of our major model components, and investigate the effects of several hyper-parameters on the performance. Finally, the discussion on explainability will be presented. Human evaluation, statistical analysis, and case study are conducted to demonstrate that KR-GCN can provide trustworthy explainability compared with baselines.

4.1 Experimental Setup

4.1.1 Datasets. We conduct extensive experiments on three widely-used public datasets about books, music, and business: Amazon-book, Last-FM, and Yelp2018. The detailed descriptions of the datasets are shown in Table 1, including the information of user-item interactions and KGs, where the KGs are introduced as complementary information to help generate more accurate and diverse recommendations. The user-item interactions are organized in the form of $u : (i_1, i_2, \ldots, i_n)$, where $i_1, i_2, \ldots, i_n$ denote the items that the user u has interacted with. These three datasets with KGs are publicly available, which are released by [46].

- **Amazon-book**: Amazon-book is a popular dataset for the book recommendation, which is selected from the dataset Amazon-review (including product reviews, product metadata, and behavior links). Amazon-book contains binary implicit feedback between users and books. If a user interacts with an item (a book), the interaction between them is 1; otherwise, the interaction is 0. The KG for this dataset is constructed by mapping the book titles in Amazon-book to the corresponding entities in Freebase [2]. The relations in the KG corresponding to Amazon-book mainly relate to books, such as *Genre*, *Topic*, and *Author*.
- **Last-FM**: Last-FM is a recommendation dataset about music, which is extracted from Last.fm online music systems. Last-FM also contains binary implicit feedback between users and music. If a user interacts with a piece of music, the interaction between them is 1; otherwise, the interaction is 0. The KG for Last-FM is also constructed by mapping the music to the corresponding entities in Freebase. The relations in the KG corresponding to Last-FM mainly relate to music, such as *Artist*, *Producer*, and *Type*.
- **Yelp2018**: Yelp2018 is extracted from Yelp challenge for **Point of Interest** (**POI**) recommendation. This dataset contains users and their check-in business data, such as restaurants, museums, parks, and so on. If a user interacts with an item (e.g., a restaurant), the interaction between them is 1; otherwise, the interaction is 0. The KG for Yelp2018 is constructed from the local business information network. The relations in the KG corresponding to Yelp2018 mainly relate to restaurants information, such as *Reservation* and *PriceRange*.

In this article, we directly adopt the dataset division results from KGAT [46] for fair comparison. For each dataset, the ratio of training and test set is 8 : 2, and the valid set is selected from the training set to tune hyperparameters. Following previous recommendation models, the negative

Table 1. Statistics of Three Benchmark Datasets

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

training instances are generated via the negative sampling strategy, which pairs each positive user-item instance with one negative item that the user has not interacted with before.

4.1.2 Baselines. To examine the effectiveness of the proposed model KR-GCN, we compare KR-GCN with the following baselines in the experiments, in which NFM, NCF, and MCRec are knowledge-agnostic models while CKE, RippleNet, KGAT, and JNSKR are knowledge-aware models.

— **NFM** [15] combines second-order feature interactions captured by FM and high-order feature interactions captured by neural networks to solve the crossover problem of sparse features. The factorization machine is considered a part of the neural network.
— **NCF** [17] replaces the inner product operation on the latent features of users and items with a neural architecture. The user-item interactions are learned by leveraging a MLP, so as to break through the limitations of the collaborative filtering methods.
— **MCRec** [19] explicitly encodes meta-paths as interactions between the users and the items, where the meta-paths are pre-defined according to the recommendation datasets. To enhance the interactions, a co-attention mechanism is used to learn representations effectively.
— **CKE** [58] extracts the KG information using the KGE method TransR [27] and then combines the KG representations with textual and visual representations to generate multi-modal representations for items.
— **RippleNet** [42] is a memory-network-like model. To predict the interaction probabilities between the users and the items, RippleNet models the propagation of users' preferences from users' historical interests along reasoning paths in the KG.
— **KGAT** [46] models the high-order relations between users and items. The embeddings of users and items are refined by recursive propagation. And an attention mechanism is used to discriminate the importance of different neighbors of the given node.
— **JNSKR** [5] applies a non-sampling optimization strategy for learning KGE. To characterize user's preferences over items, JNSKR encodes subgraphs via the knowledge-aware attentive neural networks.

4.1.3 Evaluation Metrics. To evaluate the recommendation performance of the proposed KR-GCN and the baseline models, we adopt the same evaluation metrics widely used in previous recommendation methods for fair comparisons. The evaluation metrics in our experiments include Recall and **Normalized Discounted cumulative gain (NDCG)** with the given cut-off value k (k = 10, 20, 40). Recall@k is the metric for calculating the percentage of the desired recommendation items ranked among the top k items, which is more concerned with the number of desired items among the top k items than the ranking. NDCG@k is designed to measure whether the ground truth appears in more advanced positions. This metric focuses more on the ranking of the desired recommendation items. The all-ranking protocol is adopted in our experiments, which ranks all items that have not been interacted with the given user.

4.1.4 Implementation Details. In our implementation, we randomly initialize the model parameters with the Xavier method [14] and set the size of the mini-batch to 1024, the learning rate to 0.001. We select the number of GCN layers from $\{2, 3, 4\}$, the corresponding layer combination coefficient α_l from $\{1/3, 1/4, 1/5\}$. The probability threshold p in the nucleus sampling strategy is selected from $\{0.86, 0.88, 0.9, 0.92, 0.94\}$. We also set the maximum hop of the path to 3, the dimension of the embeddings to 128 in the GCN and LSTM path encoder, and the unit number of the LSTM to 256, the training epoch to 1,000. The probabilities of the triplets within every single path are calculated by TransH, TransE, and DistMult, in which the dimension of the embeddings is fixed to 200. The L_2 regularization coefficient λ is selected from {1e-5, 1e-4, 1e-3, 1e-2, 1e-1}. For the baseline MCRec, following the setting in the work of KGAT [46], we manually define several types of meta-paths for Amazon-book dataset, such as user-book-user-book, user-book-author-book, and user-book-genre-book. The types of entities are obtained by using the relations *author* and *genre* in the KG, such as the triples $(e_1, author, a_1)$ and $(e_1, genre, g_1)$.

4.2 Performance Comparison and Analysis

In the experiments, we compare the proposed model KR-GCN with several baselines on three real-world datasets, so as to evaluate the effectiveness of KR-GCN as a whole. Then, we conduct an ablation study to discuss the effectiveness of our major model components, including users' historical interaction items, path-level attention mechanism, GCN propagation layers, and selected paths. Finally, we investigate the effect of several hyper-parameters on recommendation performance.

4.2.1 Comparison with Baselines. Tables 2 and 3 present the performance comparison results of the proposed model KR-GCN and several baselines on three datasets. Specifically, KR-GCN is the proposed model that utilizes nucleus sampling, and KR-GCN (top-k) is a variant of KR-GCN, which uses top-k sampling. KR-GCN (random) is the variant that samples k paths as input randomly. KR-GCN (TransE) and KR-GCN (DistMult) are also variants of KR-GCN, which utilize TransE and DistMult as triplet scoring functions. The major observations from the experimental results are summarized as follows:

The proposed KR-GCN consistently outperforms the baselines and the variants by a margin on three datasets. The performance comparison results demonstrate the high effectiveness of KR-GCN. We attribute it to the fact that KR-GCN alleviates the challenges in the existing models.

Among all baselines, knowledge-aware models achieve better performance compared with knowledge-agnostic models in most cases, which indicates the usefulness of incorporating KGs into recommender systems to enhance recommendation performance. For knowledge-agnostic models, NFM performs better than NCF on Amazon-book and Yelp2018, because NFM combines the second-order feature interactions captured by FM and the higher-order feature interactions captured by neural networks, which can alleviate the problem of sparse features. The performance of the path-based model MCRec is limited by the design of the meta-paths. Compared with knowledge-agnostic models, knowledge-aware models usually generate more accurate and diverse recommendation results by introducing auxiliary knowledge about items. The knowledge-aware models KGAT and JNSKR yield better performance than CKE and RippleNet. This is because that CKE ignores the global structure information of the graph, while KGAT captures the high-order relations between users and items via a KG attention network. JNSKR improves recommendation performance by applying an efficient non-sampling optimization strategy for learning KGE.

KR-GCN performs better than the GCN-based model KGAT, indicating that the recommendation performance improvements of the proposed model KR-GCN are not just due to the high performance of GCN. JNSKR might achieve good performance on Amazon-book and Yelp2018,

Table 2. Performance Comparisons of KR-GCN and Baselines on Amazon-Book and Yelp2018

	Models	Recall@10	Recall@20	Recall@40	NDCG@10	NDCG@20	NDCG@40
Amazon-book	NFM	0.0891	0.1366	0.1975	0.0723	0.0913	0.1152
	NCF	0.0874	0.1319	0.1924	0.0724	0.0895	0.1111
	MCRec	–	0.1113	–	–	0.0783	–
	CKE	0.0875	0.1343	0.1946	0.0705	0.0885	0.1114
	RippleNet	0.0883	0.1336	0.2008	0.0747	0.0910	0.1164
	KGAT	0.1017	0.1489	0.2094	0.0814	0.1006	0.1225
	JNSKR	0.1056	0.1558	0.2178	0.0842	0.1068	0.1271
	KR-GCN(top-k)	0.1058	0.1573	0.2196	0.0863	0.1076	0.1296
	KR-GCN(random)	0.1016	0.1493	0.2098	0.0809	0.1001	0.1228
	KR-GCN(TransE)	0.1074	0.1582	0.2186	0.0874	0.1090	0.1305
	KR-GCN(DistMult)	0.1095	0.1603	0.2203	0.0885	0.1107	0.1319
	KR-GCN	**0.1117**	**0.1635**	**0.2246**	**0.0892**	**0.1116**	**0.1332**
	improvement %	5.7%	4.9%	3.1%	5.9%	4.5%	4.7%
Yelp2018	NFM	0.0396	0.0660	0.1082	0.0603	0.0810	0.1094
	NCF	0.0389	0.0653	0.1060	0.0603	0.0802	0.1087
	CKE	0.0399	0.0657	0.1074	0.0608	0.0805	0.1091
	RippleNet	0.0402	0.0664	0.1088	0.0613	0.0822	0.1097
	KGAT	0.0418	0.0712	0.1128	0.0630	0.0867	0.1129
	JNSKR	0.0456	0.0749	0.1209	0.0687	0.0917	0.1211
	KR-GCN(top-k)	0.0461	0.0758	0.1213	0.0697	0.0921	0.1220
	KR-GCN(random)	0.0420	0.0715	0.1128	0.0638	0.0869	0.1134
	KR-GCN(TransE)	0.0473	0.0767	0.1249	0.0709	0.0937	0.1235
	KR-GCN(DistMult)	0.0479	0.0805	0.1281	0.0714	0.0947	0.1248
	KR-GCN	**0.0491**	**0.0813**	**0.1297**	**0.0725**	**0.0963**	**0.1267**
	improvement %	7.6%	8.5%	7.2%	5.5%	5.0%	4.6%

Best results are in bold.

Table 3. Performance Comparisons of KR-GCN and Baselines on Last-FM

Models	NFM	CKE	RippleNet	KGAT	KR-GCN (top-k)	KR-GCN (random)	KR-GCN (TransE)	KR-GCN (DistMult)	KR-GCN
Recall@20	0.0829	0.0736	0.0791	0.0870	0.0876	0.0872	0.0880	0.0886	**0.0892**
NDCG@20	0.1214	0.1184	0.1238	0.1325	0.1332	0.1327	0.1338	0.1349	**0.1353**

Best results are in bold.

but lead to low efficiency as well. The negative sampling strategy is adopted in our KR-GCN in the training process, and the non-sampling strategy could also be adopted in the follow-up work to further improve the recommendation performance.

KR-GCN (top-k) is a comparison model with top-k sampling; that is, the number of selected paths in the path selection module is fixed. The performance of KR-GCN is better than that of KR-GCN (top-k), illustrating that adaptive path selection has advantages over static path selection. KR-GCN (random) is a variant model that samples k paths randomly. KR-GCN (random) performs worse than KR-GCN and KR-GCN (top-k), indicating the importance and the effectiveness of the path selection module, i.e., verifying the high quality and usefulness of the extracted paths in KR-GCN. This also illustrates that the irrelevant paths will induce error propagation and further impact recommendation performance, which is consistent with the motivation of our article. KR-GCN uses the KGE method TransH to calculate the scores of triplets within the multi-hop paths in the path selection module, while KR-GCN (TransE) and KR-GCN (DistMult) use TransE and DistMult as triplet scoring functions, respectively. KR-GCN outperforms KR-GCN (TransE) and KR-GCN (DistMult), indicating that TransH is a more appropriate choice than TransE and DistMult for the recommendation task in calculating the scores of triplets within the multi-hop reasoning paths. We attribute this to the fact that TransH can deal with the one-to-many,

Table 4. Effects of Historical Interaction Items and Path-Level Attention

Models	Amazon-book		Last-FM		Yelp2018	
	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
w/o HI	0.1548	0.1048	0.0879	0.1329	0.0735	0.0903
w/o Att	0.1439	0.0993	0.0809	0.1257	0.0698	0.0859
w/o HI&Att	0.1422	0.0987	0.0783	0.1230	0.0664	0.0832
KR-GCN	0.1635	0.1116	0.0892	0.1353	0.0813	0.0963

many-to-one, and many-to-many relation patterns in the graph, and these relation patterns widely exist in the user-item interactions and KG, especially in the user-item interactions. Although TransE and DistMult can also be used to calculate multi-hop path scores, TransE cannot model the relation patterns mentioned above very well, and DistMult cannot model symmetric relations in the paths. For example, two triplets (*A Tale of Two cities*, *WrittenBy*, *Charles Dickens*) and (*Charles Dickens*, *WrittenBy*, *A Tale of Two Cities*) within the path would be calculated as the same scores via DistMult, and the second triplet is clearly wrong.

Compared with the best performance results in baselines (i.e., JNSKR in Amazon-book, Yelp2018, and KGAT in Last-FM), KR-GCN improves by 4.9%, 8.5%, and 2.5% measured by Recall@20, and improves by 4.4%, 5.0%, and 2.1% measured by NDCG@20. We attribute the superior recommendation performance of the proposed KR-GCN to leveraging graph structure information as well as semantic information, and selecting reasoning paths that are more helpful for predicting user preferences. To verify that KR-GCN outperforms baselines significantly, we conduct significance testing, and differences between KR-GCN and baselines are significant ($p < 0.05$) using the t-test. Specifically, to perform significance testing, we assume that it is no difference between the experimental results of baselines and KR-GCN. The experimental results are compared to calculate the p-value between each baseline and KR-GCN. Finally, we find that $p < 0.05$ for every two groups, so the null hypothesis is rejected,. i.e., there is a significant difference between the experimental results of KR-GCN and baselines.

Among all these models, path-based models MCRec, RippleNet, and the proposed KR-GCN can provide explainability by providing reasoning paths for the recommendation. MCRec explicitly encodes meta-paths as the interactions between user-item pairs, in which the diversity of reasoning paths (i.e., the explanations for the recommendation results) is limited because of the design of the meta-paths. RippleNet tracks the paths from the userâĂŹs historical interactions to the items in the KG, and then discovers possible reasoning paths as the explanations. However, the collaborative information is ignored in RippleNet, for example, the multi-hop path $u_1 \xrightarrow{Interact} i_1 \xrightarrow{Interact} u_2 \xrightarrow{Interact} i_2$ is not considered and utilized to reason user preferences. Different from MCRec and RippleNet, KR-GCN first integrates the user-item interactions and KG (i.e., user collaboration information and auxiliary knowledge about items) into a heterogeneous graph. Then, KR-GCN extracts reasoning paths from the heterogeneous graph, and utilizes an attention mechanism to discriminate the effects of different multi-hop paths between users and items, thus providing trustworthy explainability. Moreover, KR-GCN prunes irrelevant paths in the path selection module to mitigate interference in the reasoning and explanations.

4.2.2 Ablation Study. To investigate the effects of different model components on the performance of KR-GCN, we first conduct an ablation study to analyze the effect of the users' historical interaction items and path-level attention mechanism. Then, we explore the components of GCN propagation layers and the selected paths on the experimental results on three datasets.

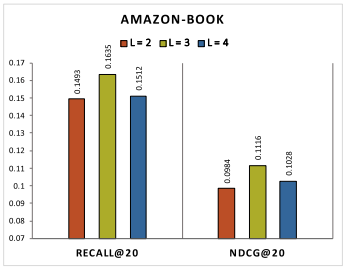
We analyze the effects of the historical interaction items and path-level self-attention mechanism and then summarize the recommendation performance comparisons in Table 4. These

two components are abbreviated as HI and Att. Removing either the historical interaction items component or path-level self-attention mechanism component degrades the recommendation performance, and the performance is worst when both components are removed. The evaluation results prove that both the historical interaction items and the path-level self-attention mechanism make contributions to infer user preferences. The component of users' historical interaction items is encoded as additional input features and combined with each path to reflect users' interests, which can also enhance the representations of the path sequences. Leveraging the path-level self-attention mechanism in KR-GCN can learn path-specific weight and discriminate the contributions of different reasoning paths between users and items, so as to capture more reasonable user interests and provide a more accurate recommendation.

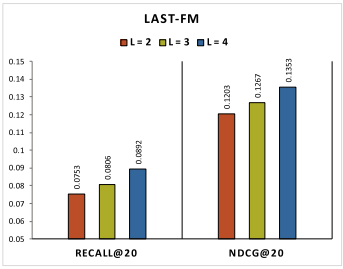
To explore the effects of components of GCN propagation layers and selected paths, we conduct experiments on different GCN propagation layers, probability threshold of triplets, and path hops, i.e., the length of paths. Figure 3 shows the recommendation performance comparisons of different GCN propagation layers, probability threshold, and path hops. The comparison results are measured by the metrics Recall@20 and NDCG@20 on the datasets Amazon-book, Last-FM, and Yelp2018. L, p, and l denote the number of propagation layers in the graph encoding module, the probability threshold in the nucleus sampling strategy, and the path hop in the path extraction module, respectively. For different GCN propagation layers and probability threshold, we can observe that when $L = 3$ and $p = 0.9$, KR-GCN obtains better recommendation performance on Amazon-book with all metrics, and when $L = 4$ and $p = 0.92$, KR-GCN obtains better performance on Last-FM and Yelp2018, as shown in Figure 3(a)–(f). An intuitive explanation is that there are more user-item interactions in Last-FM and Yelp2018 than in Amazon-book. More GCN propagation layers and higher path probability threshold represent richer training information. However, the recommendation performance does not grow with the number of GCN propagation layers and path probability threshold because more GCN propagation layers and higher path probability threshold can also cause over-smooth and introduce interference information. For different path hops, we compare the model performance when $l = 3$ and $l = 5$. The main reason is that the model cannot utilize path information when $l = 1$, when $l = 2$ or $l = 4$, path extraction is unreasonable, and when $l > 5$, the path extraction and selection is very inefficient. When $I = 3$, the path types include: $u - i - u/e - i$, and when $I = 5$, the path types include: $u - i - u/e - i - u/e - i$ and $u - i - e - e/i - e - i$. As shown in Figure 3(g)–(i), when $l = 3$, the performance is better than when $l = 5$ on three datasets. We attribute this to the fact that long paths enhance semantic information but also weaken the associations between users and items.

4.2.3 Hyper-Parameter Study. KR-GCN involves a number of hyper-parameters. In this subsection, we investigate the sensitivity of hyper-parameters on the recommendation performance and report the experimental results on two datasets in Figures 4 and 5.

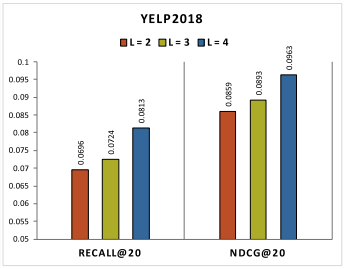
— **Dimension of the embeddings in the GCN.** We first test the effects of the embedding dimension d_1 in the GCN on the performance. GCN is utilized to encode the heterogeneous graph that integrates the user-item interactions and the KG. Therefore, parameter setting in the GCN has a key effect on recommendation performance. Figure 4 shows the experimental results of different embedding dimension in the GCN on Amazon-book, Last-FM, and Yelp2018, which are evaluated by the metrics Recall@20 and NDCG@20. The experimental results demonstrate that with the growth of the embedding dimension, the recommendation performance raises first and then starts to drop. We can see that KR-GCN achieves the best recommendation performance when the embedding dimension d_1 is set to 128. This indicates that the high embedding dimension can enhance node representations and improve model



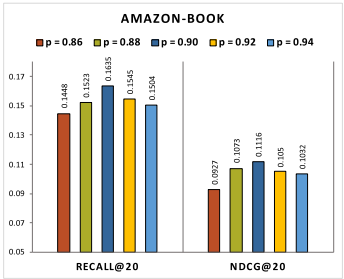
(a) Performance comparisons of different GCN propagation layers on Amazon-book.



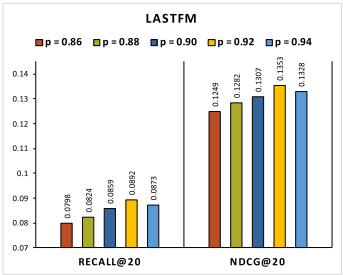
(b) Performance comparisons of different GCN propagation layers on Last-FM.



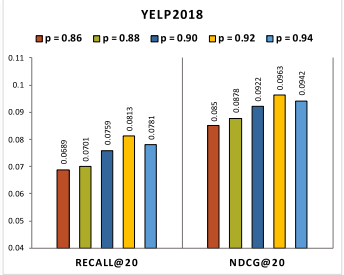
(c) Performance comparisons of different GCN propagation layers on Yelp2018.



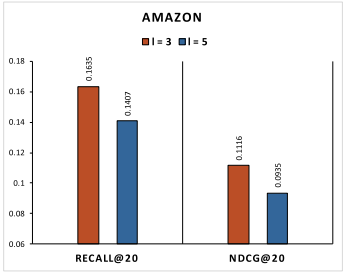
(d) Performance comparisons of different path probability threshold on Amazon-book.



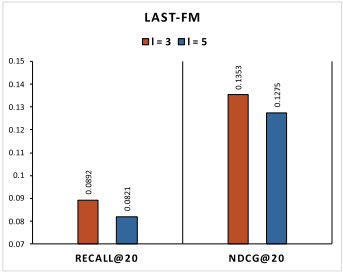
(e) Performance comparisons of different path probability threshold on Last-FM.



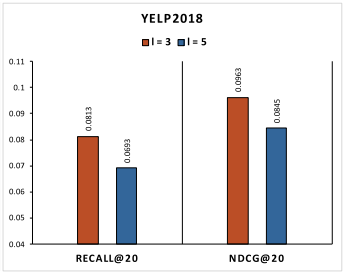
(f) Performance comparisons of different path probability threshold on Yelp2018.



(g) Performance comparisons of different path hops on Amazon-book.



(h) Performance comparisons of different path hops on Last-FM.



(i) Performance comparisons of different path hops on Yelp2018.

Fig. 3. Performance comparisons of different GCN propagation layers, path probability thresholds, and path hops on three datasets.

performance, but the higher dimension is not always better, because additional dependencies are also introduced.

— **Dimension of the embeddings in the TransH.** TransH is used to calculate the scores of multi-hop paths in the path selection module. The parameters in TransH affect the selection of reasoning paths, and then affect the model performance. To explore the effects of the parameters in TransH on the model performance, we start experiments with different embedding dimension d_2 in the TransH to check on its influence. The experimental results of different embedding dimension in the TransH on Amazon-book, Last-FM, and Yelp2018 are shown in Figure 5. Recall@20 and NDCG@20 are also used as evaluation metrics. From the experimental results, we can find that the embedding dimension $d_2 = 200$ is the best choice for KR-GCN. Similar to the conclusion of the embedding dimension in GCN, the high

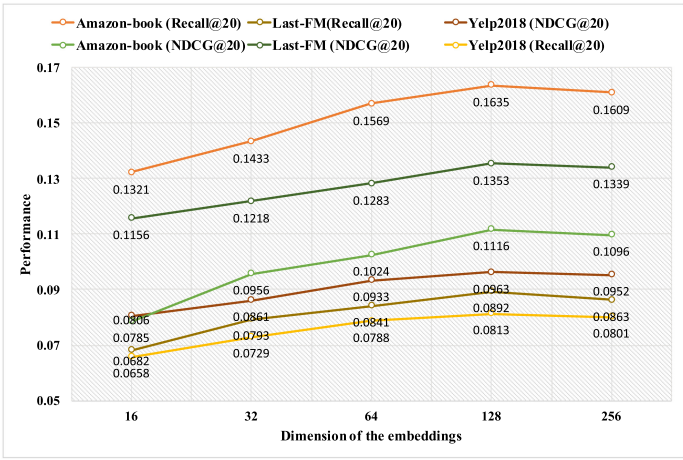


Fig. 4. The experimental results of different embedding dimension in the GCN on Amazon-book, Last-FM, and Yelp2018.

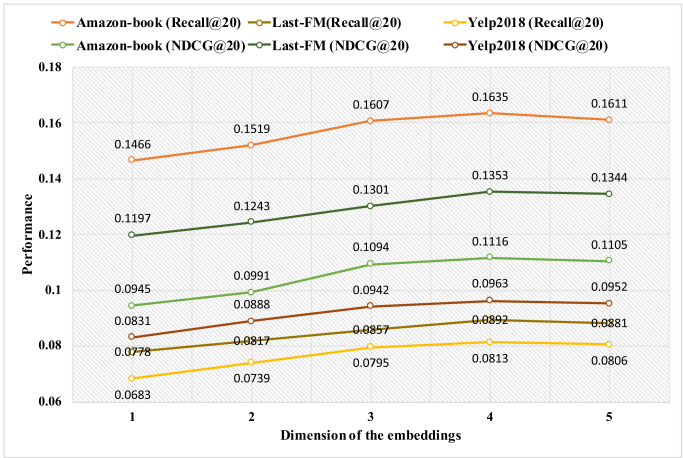


Fig. 5. The experimental results of different embedding dimension in the TransH on Amazon-book, Last-FM, and Yelp2018.

embedding dimension can improve model performance, but the higher dimension is not always better, because the higher dimension will lead to overfitting of the model.

4.3 Discussion on Explainability

In this subsection, we conduct the human evaluation, statistical analysis and case study to discuss the explainability provided by KR-GCN and comparison models.

4.3.1 Human Evaluation on Explainability. To verify that the proposed model can provide more trustworthy explainability for recommended items, the reasoning paths (i.e., recommendation explanations) that provided by models are evaluated manually. The evaluation metrics include *Relevance* and *Diversity*. *Relevance* is used to evaluate whether the explanations are closely related to the recommended items, for example, for the item *Hamlet*, taking *Romeo and Juliet* as an

explanation is more relevant than *English*. *Diversity* is used to evaluate whether the explanations include multiple relationships between every user-item pair. We define the *Diversity* of the path p between the given user-item pair as follows:

$$Diversity(p) = \alpha(N(R_p) + N(type(E_p))), \tag{24}$$

where $N(R_p)$ is the number of the relations within the path p, $type(E_p)$ is the types of the entities within the path p, $N(type(E_p))$ is the number of these types, and α is a scale parameter of the direct proportional function. The diversity of explanation is mainly related to the number of relations and the number of entity types. For example, given the path p_1 and p_2:

$$p_1 : u_{165} \xrightarrow{Interact} BlindLake \xrightarrow{Interact} u28177 \xrightarrow{Interact} Flashforward, \tag{25}$$

$$p_2 : u_{165} \xrightarrow{Interact} BlindLake \xrightarrow{Genre} ScienceFiction \xrightarrow{InGenre} Flashforward, \tag{26}$$

where the number of relations within the path p_1 and p_2 is 1 and 3, and the number of entity types is 2 and 3, respectively. So the path p_1 will be scored lower than the path p_2 and more likely to be the explanation for recommendation.

On one hand, when the length of the path is too long, the path extraction and selection will be very inefficient, and the associations between users and items will also be weakened. On the other hand, the diversity of the path is correlated with the number of path hops. How to make a good balance within the length of the paths, the diversity of the paths, and the explainability is a key issue.

We first determine the number of path hops. Because we mainly focus on the recommendation performance and explainability in this article, where the performance should be considered a priority. And, the number of path hops is directly related to the recommendation performance. In this article, the best choice for the number of hops is three hops on three datasets. After the number of path hops is determined, we focus on the explainability of the model, i.e., the reasoning paths provided by the model. Because the relevance of the reasoning paths will affect the model performance, we first focus on the relevance of explanations, which is guaranteed by the path selection module and the attention mechanism. Then, we focus on the diversity of explanations. Specifically, when the number of paths hops is set, our goal is to make the reasoning paths include more entity types and relations. In this article, we do not use the meta-path patterns to mine the reasoning paths, so the limitations of entity type can be alleviated to a certain extent. To conclude, our goal is to focus on the explainability of the recommendation model under the premise that the recommendation performance is considerable; that is, we first focus on the path length, then the relevance of the explanations, and finally, the diversity of the explanations.

To evaluate the explainability, we randomly sample 100 user-item pairs, and output reasoning paths provided by models. Specifically, five reasoning paths with higher attention scores between every user-item pair are selected for comparison. Then, we select 10 raters who have the machine learning experience to evaluate the relevance and diversity of these reasoning paths. The evaluation scores range from 0 to 10 for both metrics. Specifically, each model output 500 (100*5) paths. We divide these paths into 5 groups, and each group contains 100 paths that correspond to 20 user-item pairs. To avoid the problem of inconsistent labeling results between different raters, each group (100 paths) is assigned to 2 raters for evaluation and the score of each path is obtained by averaging the scores provided by two raters. Finally, the scores of all the comparison paths are obtained.

Among the baselines, the path-based models MCRec and RippleNet have the ability to provide explainability. The explainability comparisons of MCRec, RippleNet, and the proposed KR-GCN on the Amazon-book are shown in Table 5. The evaluation results prove that our KR-GCN can provide

Table 5. The Explainability Comparisons of MCRec, RippleNet, and KR-GCN on Amazon-Book

Models	Amazon-book	
	Relevance	Diversity
MCRec	7.21	6.43
RippleNet	6.82	7.16
KR-GCN	7.61	7.85

more relevant and diverse explanations compared with MCRec and RippleNet. We attribute this to solving the error propagation problem of the paths and integrating the user-item interactions and KG. The former improves relevance, and the latter improves diversity. In terms of relevance, KR-GCN filters the irrelevant paths between user-item pairs to solve the problem of error propagation, so the final reasoning paths are more relevant to the recommended items. In terms of diversity, KR-GCN constructs a heterogeneous graph including user collaboration information and auxiliary knowledge about items. Nodes and relations within the final explanation paths are not limited by the types of user/item/entity/relations, that is, the diversity of explanations is not limited by data. The diversity of reasoning paths in McRec is limited by designing meta-paths for KGs, and RippleNet ignores user collaborative information, so the diversity of reasoning paths is also limited. Compared with McRec and RippleNet, KR-GCN can provide more diverse explanations.

4.3.2 Statistical Analysis. To further demonstrate the explanations, we compare reasoning subgraphs provided by the proposed model KR-GCN and two baselines MCRec and RippleNet, where the reasoning subgraphs are composed of the reasoning paths and have more complicated structures than the reasoning paths. Specifically, five reasoning paths with higher attention scores between every user-item pair are selected and formed into reasoning subgraphs for comparison. The subgraphs provided by the three models are compared by computing the similarity of their topological sequences, where the similarity of each two topological sequences is measured via Levenshtein distance [30]. Figure 6 shows the statistical results of the subgraph similarity between KR-GCN and two baselines (i.e., MCRec and RippleNet). The statistical results demonstrate that a large number of reasoning subgraphs provided by KR-GCN and by baselines (especially RippleNet) have high similarity, because many reasoning subgraphs are in the similarity interval of 50%–100%. This indicates that the reasoning subgraphs provided by KR-GCN have some overlap with those provided by baselines; that is, most of the reasoning paths provided by baselines can be covered by KR-GCN. Furthermore, compared with the baselines, KR-GCN performs higher relevance because of the attention mechanism, and the diversity is not limited. In general, we believe that KR-GCN can generate higher-quality reasoning subgraphs to provide explanations for the recommendation.

4.3.3 Case Study on Explainability. We present an example from Amazon-book to demonstrate the explainability of MCRec, RippleNet, and the proposed KR-GCN. Figure 7 shows the interactions between the user u_{165} and the item *Flashforward* that are explored by three models. For all models, five reasoning paths with higher attention scores are selected and formed into reasoning subgraphs for comparison and display. We can find that KR-GCN provides more types of relationships between the user u_{165} and the item *Flashforward* than MCRec and RippleNet. The main reason is that McRec only defines a limited number of meta-paths, and RippleNet ignores the user collaboration information in the graph. As shown in Figure 7, three paths in MCRec only contain the user-item interactions, i.e., the paths p_3, p_4, and p_5, and all paths in RippleNet only contain the KG information. In addition, the reasoning paths are not sorted and pruned in MCRec and RippleNet, the final reasoning paths/subgraphs might include some low-quality and irrelevant

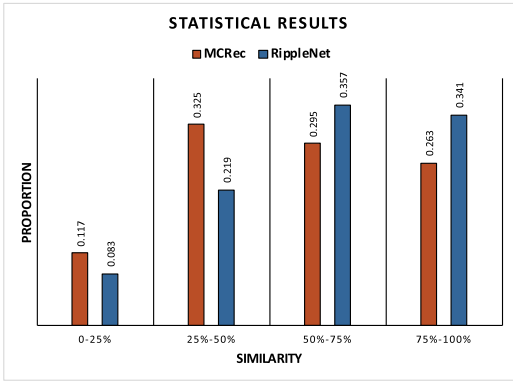


Fig. 6. The statistical results of the subgraph similarity between baselines (i.e., MCRec and RippleNet) and KR-GCN. The horizontal axis shows the similarity interval, and the vertical axis shows the proportion of the number of subgraphs in a certain similarity interval to the number of all subgraphs.

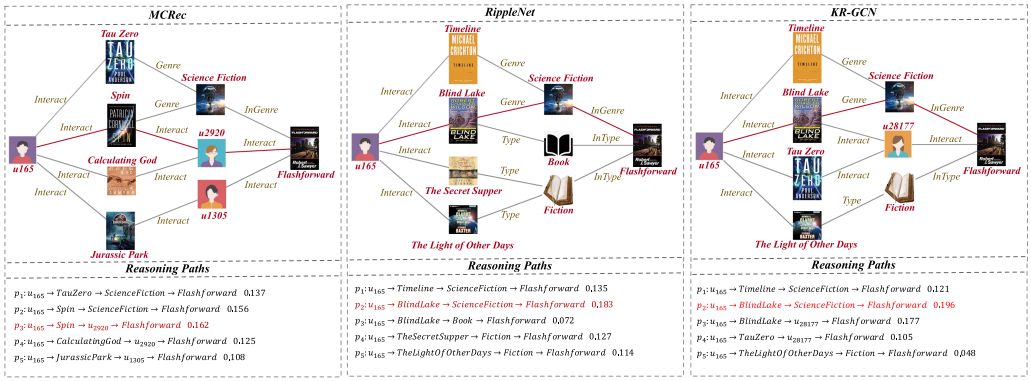


Fig. 7. Real example from Amazon-book. The interactions between the user u_{165} and the item *Flashforward* are explored by MCRec, RippleNet, and KR-GCN. The reasoning paths in red indicate the explanations for recommendation.

paths, such as the path $u_{165} \xrightarrow{Interact} Blind\ Lake \xrightarrow{Type} Book \xrightarrow{InType} Flashforward$ in RippleNet. For KR-GCN, the path p_2 obtains the highest attention score as shown in Figure 7. We can summarize the reason for recommending *Flashforward* to u_{165} as follows: *the user* u_{165} *likes the book Blind Lake, whose genre is science fiction, and Flashforward is also a science fiction.*

5 CONLUSION

In this article, we propose a novel method named KR-GCN for the explainable recommendation. We integrate user-item interactions and KG into a heterogeneous graph and encode the graph with the GCN to improve the recommendation performance. To cope with the error propagation in the graph, we develop a transition-based method to score triplets within multi-hop paths between every user-item pair and utilize nucleus sampling to adaptively select triplets. To provide trustworthy explainability, we introduce a path-level self-attention mechanism to discriminate the contributions of different selected paths and predict the interaction probability, and the path with the highest weight is provided as the explanation for the recommendation. Extensive experiments

on three real-world datasets show that the proposed KR-GCN achieves improvements over state-of-the-art baselines. And, human evaluation proves that KR-GCN can provide more trustworthy explainability compared with baselines.

Future research directions include (1) exploring other ways to take advantage of both GCN-based methods and path-based methods, and (2) trying to apply path selection strategy in other tasks, such as link prediction. On one hand, GCN-based methods have proved to be successful in modeling the high-order relations in KGs to provide better recommendation. And the state-of-the-art performance on public datasets is mostly achieved by GCN-based methods. Path-based methods can make full use of reasoning paths to model the propagation of users' interests, so as to provide more accurate and explainable recommendation. So exploring ways to take advantage of both can provide high-performance and user-friendly recommendation. On the other hand, the path selection strategy is important for path-related tasks, not just for recommendation task. The path selection strategy aims at finding more potential reasoning paths and avoid introducing interference information, which can also alleviate the problem of error propagation in other path-related tasks.

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] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD*. 1247–1250.

[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of the Advances in Neural Information Processing Systems 26: The 27th Annual Conference on Neural Information Processing Systems*. 2787–2795.

[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 *Proceedings of the 2019 World Wide Web Conference on World Wide Web,WWW*. 151–161.

[5] Chong Chen, Min Zhang, Weizhi Ma, Yiqun Liu, and Shaoping Ma. 2020. Jointly non-sampling learning for knowledge graph enhanced recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR*. 189–198.

[6] Hongxu Chen, Yicong Li, Xiangguo Sun, Guandong Xu, and Hongzhi Yin. 2021. Temporal meta-path guided explainable recommendation. In *Proceedings of the WSDM'21, The Fourteenth ACM International Conference on Web Search and Data Mining, 2021*. ACM, 1056–1064.

[7] Xu Chen, Kun Xiong, Yongfeng Zhang, Long Xia, Dawei Yin, and Jimmy Xiangji Huang. 2020. Neural feature-aware recommendation with signed hypergraph convolutional network. *ACM Transactions on Information Systems* 39, 1 (2020), 8:1–8:22.

[8] Zhongxia Chen, Xiting Wang, Xing Xie, Tong Wu, Guoqing Bu, Yining Wang, and Enhong Chen. 2019. Co-Attentive multi-task learning for explainable recommendation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI 2019*, Sarit Kraus (Ed.). 2137–2143.

[9] Zhiyong Cheng, Xiaojun Chang, Lei Zhu, Rose Catherine Kanjirathinkal, and Mohan S. Kankanhalli. 2019. MMALFM: Explainable recommendation by leveraging reviews and images. *ACM Transactions on Information Systems* 37, 2 (2019), 16:1–16:28.

[10] 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 and Data Mining, KDD*. 2478–2486.

[11] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Yihong Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *Proceedings of the World Wide Web Conference, WWW 2019*. 417–426.

[12] Zuohui Fu, Yikun Xian, Ruoyuan Gao, Jieyu Zhao, Qiaoying Huang, Yingqiang Ge, Shuyuan Xu, Shijie Geng, Chirag Shah, Yongfeng Zhang, and Gerard de Melo. 2020. Fairness-Aware explainable recommendation over knowledge graphs. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2020*. 69–78.

[13] Jingyue Gao, Xiting Wang, Yasha Wang, and Xing Xie. 2019. Explainable Recommendation through Attentive Multi-View Learning. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence, AAAI 2019*. 3622–3629.

[14] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics, AISTATS 2010 (JMLR Proceedings, Vol. 9)*. 249–256.

[15] 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, SIGIR*. 355–364.

[16] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong 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, SIGIR*. 639–648.

[17] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web, WWW*. 173–182.

[18] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *Proceedings of the 8th International Conference on Learning Representations, ICLR 2020*.

[19] Binbin Hu, Chuan Shi, Wayne Xin Zhao, and Philip S. Yu. 2018. Leveraging meta-path based Context for Top-N recommendation with a neural co-attention model. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD*. 1531–1540.

[20] Xiaowen Huang, Quan Fang, Shengsheng Qian, Jitao Sang, Yan Li, and Changsheng Xu. 2019. Explainable interaction-driven user modeling over knowledge graph for sequential recommendation. In *Proceedings of the 27th ACM International Conference on Multimedia, MM 2019*. 548–556.

[21] Kingma, Diederik, and Ba Jimmy. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3th International Conference on Learning Representations (ICLR'15)*.

[22] Thomas N. Kipf and Max Welling. 2016. Variational graph auto-encoders. *CoRR* abs/1611.07308 (2016).

[23] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations, ICLR*.

[24] Chongxuan Li, Max Welling, Jun Zhu, and Bo Zhang. 2018. Graphical generative adversarial networks. In *Proceedings of the Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems, NeurIPS 2018*. 6072–6083.

[25] Mengmeng Li, Tian Gan, Meng Liu, Zhiyong Cheng, Jianhua Yin, and Liqiang Nie. 2019. Long-tail hashtag recommendation for micro-videos with graph convolutional network. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019*. 509–518.

[26] Xueqi Li, Wenjun Jiang, Weiguang Chen, Jie Wu, Guojun Wang, and Kenli Li. 2020. Directional and explainable serendipity recommendation. In *Proceedings of the WWW'20: The Web Conference, 2020*. 122–132.

[27] 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 29th AAAI Conference on Artificial Intelligence Learning*. 2181–2187.

[28] Danyang Liu, Jianxun Lian, Zheng Liu, Xiting Wang, Guangzhong Sun, and Xing Xie. 2021. Reinforced anchor knowledge graph generation for news recommendation reasoning. In *Proceedings of the KDD'21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1055–1065.

[29] 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 *Proceedings of the World Wide Web Conference, WWW 2019*. 1210–1221.

[30] Gonzalo Navarro. 2001. A guided tour to approximate string matching. *ACM Computing Surveys* 33, 1 (2001), 31–88.

[31] Ruihong Qiu, Zi Huang, Jingjing Li, and Hongzhi Yin. 2020. Exploiting cross-session information for session-based recommendation with graph neural networks. *ACM Transactions on Information Systems* 38, 3 (2020), 22:1–22:23.

[32] Yanru Qu, Ting Bai, Weinan Zhang, Jian-Yun Nie, and Jian Tang. 2019. An end-to-end neighborhood-based interaction model for knowledge-enhanced recommendation. *CoRR* abs/1908.04032 (2019).

[33] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. In *Proceedings of the Unpublished Manuscript*. Retrieved from https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

[34] Rahul Ragesh, Sundararajan Sellamanickam, Arun Iyer, Ramakrishna Bairi, and Vijay Lingam. 2021. HeteGCN: Heterogeneous graph convolutional networks for text classification. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 860–868.

[35] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*. 452–461.

[36] Weiping Song, Zhijian Duan, Ziqing Yang, Hao Zhu, Ming Zhang, and Jian Tang. 2019. Explainable knowledge graph-based recommendation via deep reinforcement learning. *CoRR* abs/1906.09506 (2019).

[37] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. 2019. Session-based social recommendation via dynamic graph attention networks. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining, WSDM 2019*. 555–563.

[38] 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.

[39] 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 and Data Mining, KDD 2019*. 1891–1899.

[40] Trouillon, Johannes Welb, Sebastian Riedel, Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the 33th International Conference on Machine Learning*. 2071–2080.

[41] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of the 6th International Conference on Learning Representations, ICLR 2018*. OpenReview.net.

[42] 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 on Conference on Information and Knowledge Management, CIKM*. 417–426.

[43] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. [n.d.]. DKN: Deep knowledge-aware network for news recommendation. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW*.

[44] 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 and Data Mining, KDD 2019*. 968–977.

[45] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. 2019. Knowledge graph convolutional networks for recommender systems. In *Proceedings of the 2019 World Wide Web Conference on World Wide Web, WWW*. 3307–3313.

[46] 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 and Data Mining, KDD*. 950–958.

[47] 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 33rd AAAI Conference on Artificial Intelligence, AAAI*. 5329–5336.

[48] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*. 1112–1119.

[49] Qitian Wu, Hengrui Zhang, Xiaofeng Gao, Peng He, Paul Weng, Han Gao, and Guihai Chen. 2019. Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems. In *Proceedings of the World Wide Web Conference, WWW 2019*. 2091–2102.

[50] Yikun Xian, Zuohui Fu, S. Muthukrishnan, Gerard de Melo, and Yongfeng Zhang. 2019. Reinforcement knowledge graph reasoning for explainable recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR*. 285–294.

[51] Xin Xin, Xiangnan He, Yongfeng Zhang, Yongdong Zhang, and Joemon M. Jose. 2019. Relational collaborative filtering: Modeling multiple item relations for recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019*. 125–134.

[52] Feng Xue, Xiangnan He, Xiang Wang, Jiandong Xu, Kai Liu, and Richang Hong. 2019. Deep item-based collaborative filtering for Top-N recommendation. *ACM Transactions on Information Systems* 37, 3 (2019), 33:1–33:25.

[53] Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the 3rd International Conference on Learning Representations, ICLR*.

[54] Deqing Yang, Zikai Guo, Ziyi Wang, Juyang Jiang, Yanghua Xiao, and Wei Wang. 2018. A knowledge-enhanced deep recommendation framework incorporating GAN-Based models. In *Proceedings of the IEEE International Conference on Data Mining, ICDM 2018*. IEEE Computer Society, 1368–1373.

[55] Deqing Yang, Zengchun Song, Lvxin Xue, and Yanghua Xiao. 2020. A knowledge-enhanced recommendation model with attribute-level co-attention. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020*. 1909–1912.

[56] Lin Bill Yuchen, Chen Xinyue, Chen Jamin, and Ren Xiang. 2019. KagNet: Knowledge-Aware graph networks for commonsense reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2829–2839.

[57] Zijie Zeng, Jing Lin, Lin Li, Weike Pan, and Zhong Ming. 2020. Next-Item recommendation via collaborative filtering with bidirectional item similarity. *ACM Transactions on Information Systems* 38, 1 (2020), 7:1–7:22.

[58] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD*. 353–362.

[59] Jiani Zhang, Xingjian Shi, Shenglin Zhao, and Irwin King. 2019. STAR-GCN: Stacked and reconstructed graph convolutional networks for recommender systems. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI 2019*. 4264–4270.

[60] Yuan Zhang, Fei Sun, Xiaoyong Yang, Chen Xu, Wenwu Ou, and Yan Zhang. 2020. Graph-based regularization on embedding layers for recommendation. *ACM Transactions on Information Systems* 39, 1 (2020), 2:1–2:27.

[61] Yufeng Zhang, Xueli Yu, Zeyu Cui, Shu Wu, Zhongzhen Wen, and Liang Wang. 2020. Every document owns its structure: Inductive text classification via graph neural networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL, 2020*. 334–339.

[62] 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, KDD*. 635–644.

[63] Jun Zhao, Zhou Zhou, Ziyu Guan, Wei Zhao, Wei Ning, Guang Qiu, and Xiaofei He. 2019. IntentGC: A scalable graph convolution framework fusing heterogeneous information for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2019*. 2347–2357.

[64] Kangzhi Zhao, Xiting Wang, Yuren Zhang, Li Zhao, Zheng Liu, Chunxiao Xing, and Xing Xie. 2020. Leveraging demonstrations for reinforcement recommendation reasoning over knowledge graphs. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR*. 239–248.

[65] Qiannan Zhu, Xiaofei Zhou, Jia Wu, Jianlong Tan, and Li Guo. 2020. A knowledge-aware attentional reasoning network for recommendation. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence, AAAI*. 6999–7006.

Received 16 April 2021; revised 2 September 2021; accepted 10 January 2022