scientific reports

Check for updates

OPEN

Knowledge graph convolutional networks with user preferences for course recommendation

Zhong Hua, Jianbai Yang✉ & Weidong Ji

With the rapid growth of the internet and online education resources, the number of massive open online courses (MOOCs) has increased dramatically, making it difficult for users to find personalized courses that meet their needs. Knowledge graphs (KGs) have been employed in recommendation systems to effectively address the issue of sparse interaction data in the MOOC scenarios for their rich semantic information. Research on KG-enhanced recommendation algorithms has found that the utilization of side information in KGs is crucial for improving accuracy. This paper introduces KGCN-UP (Knowledge Graph Convolutional Networks with User Preferences), a novel model for predicting the likelihood of a user interacting with a course based on user preferences and item relationships within a knowledge graph. The KGCN-UP model consists of two key modules. First, the user preference propagation module refines user preferences by exploring relational chains in the knowledge graph and dynamically adjusting attention to improve user representation. Second, the item neighbor enhancement module enhances item representations by aggregating semantic relationships and assigning attention weights based on the type of relationship between entities. Together, these components address the challenge of data sparsity and improve the quality of recommendations by leveraging high-order structural and semantic information. Empirical results on a real-world dataset demonstrate that KGCN-UP significantly outperforms existing state-of-the-art recommendation models in terms of accuracy.

Keywords Personalized recommendation, Knowledge Graph, Graph Convolution Networks.

The rapid development of online education platforms has led to a variety of massive open online courses (MOOCs), including Chinese University MOOCs, Xuetang Online, and E-Cloud Classes. These platforms offer rich online learning resources, attract an increasing number of users, and provide fair and open educational opportunities for both college students and lifelong learners.

In the vast landscape of online education resources, the way users select courses that meet their specific knowledge needs is crucial, as it directly impacts the completion and dropout rates of learners[1]. As illustrated in Fig. 1, the average interaction of user-item (learner-course) for online courses is significantly lower compared to traditional recommendation scenarios, such as those for books, food, and movies. Given that course learning is often a lengthy and somewhat tedious process, this discrepancy has led to a contradiction between the proliferation of online courses and the noticeable lack of interaction. To address the issue of ''information overload", providing users with personalized courses becomes particularly important[2,47]. In light of this challenge, the recommendation algorithms used by online education platforms have increasingly become a focal point in educational research, attracting significant attention from scholars. Traditional recommendation algorithms typically rely on using side information to enhance the accuracy and effectiveness of recommendations[50]. For example, social network data[3], image features[4], contextual information[5,51], and item attributes[6,46] are widely utilized to boost the performance of recommendation systems. The aforementioned recommendation methods, although they improve recommendation performance to some extent, still struggle to effectively address key issues such as data sparsity and the difficulty of providing explainable recommendations in the MOOC environment. In contrast to these information, knowledge graph (KG) inherently offers the advantage of revealing connections between learners and course entities, thus providing implicit feedback on diverse intentions of users. Consequently, many scholars have introduced KG into their recommendation tasks[7,48,49].

Through research on recommendation algorithms based on KG, improving the utilization of relational information within KGs has become a key factor in enhancing model performance. Based on these

College of Computer Science and Information Engineering, Harbin Normal University, Harbin 150025, China. ✉email: yangjianbai@hrbnu.edu.cn

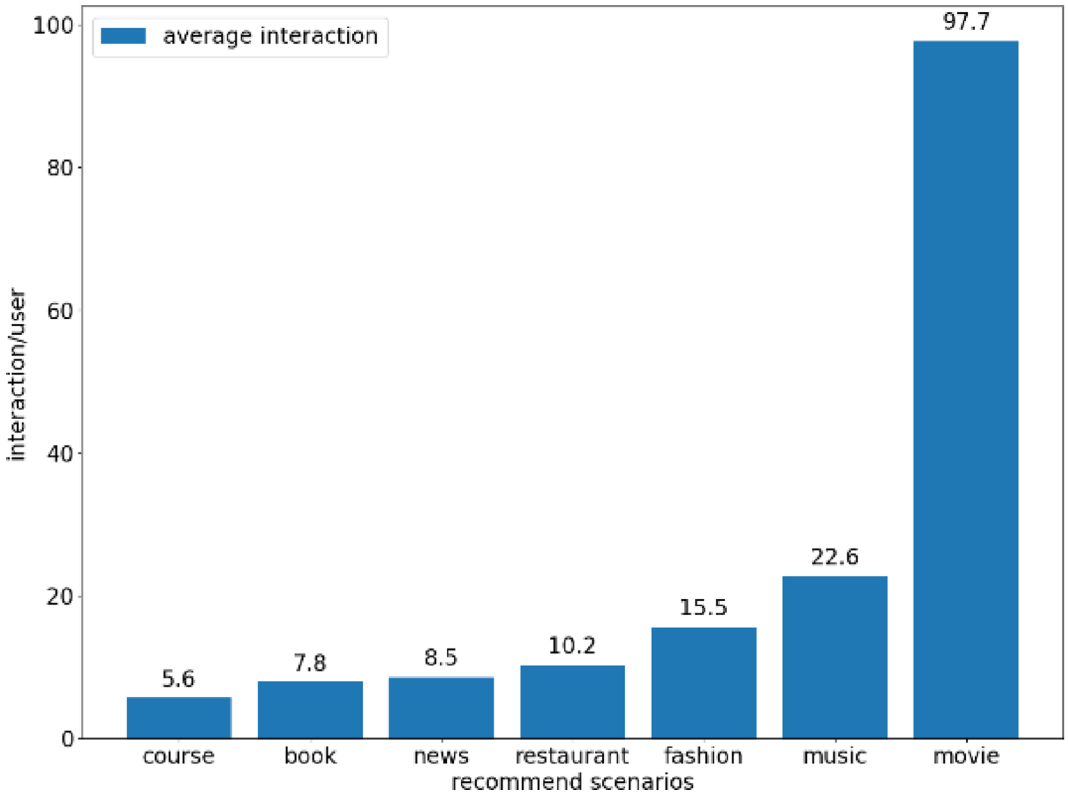

Fig. 1. Interaction-avg shows the average interaction count in the MOOCs dataset in comparison with the four other commonly used datasets.

considerations, we propose a model named KGCN-UP, short for Knowledge Graph Convolutional Networks with User Preferences, as illustrated in Fig. 2.

This model traverses all courses with which the user has historically interacted and propagates through various relationship chains in the KG to get courses that the user is likely to be interested in. We define this process as the user preference propagation module. Similarly, the model starts with candidate courses and traverses related courses adjacent to these, enhancing the representation of the candidate courses. This forms what we call the item neighbor enhancement module. This approach not only enriches the user representation but also effectively utilizes the relational chain information in the KG, thereby addressing the issue of ''information overload" and improving the model's performance.

The main contributions of this work are summarized as follows:

1) We systematically analyze the sparsity of user interaction data in MOOCs scenarios. Based on these characteristics, we propose a model named KGCN-UP that enhances user and item representations to predict course click probabilities, addressing data sparsity in MOOCs.
2) To address the sparsity problem, a novel user representation method is introduced to propagate user preferences through the knowledge graph by capturing relevant entities connected to previously interacted courses.
3) To improve the use of heterogeneous information in the knowledge graph, we design a novel approach to item representation. It traverses entities related to the candidate item and applies message passing to aggregate neighbor information for the final representation.
4) Extensive experiments demonstrate that KGCN-UP outperforms leading benchmark models on educational, book, and music recommendation tasks, with ablation studies confirming the effectiveness of both key model components.

The structure of this paper is organized as follows: "PROBLEM DEFINITION" section introduces the purpose of the recommendation task and related definitions. "RELATED WORKS" section introduces the relevant research on recommendation models based on KG. Afterwards, the proposed method is introduced in detail in "METHODS" section. Experimental results and related discussions are then presented in "EXPERIMENT". we summarize the findings of this paper and outline our plans for future research in "CONCLUSION" section.

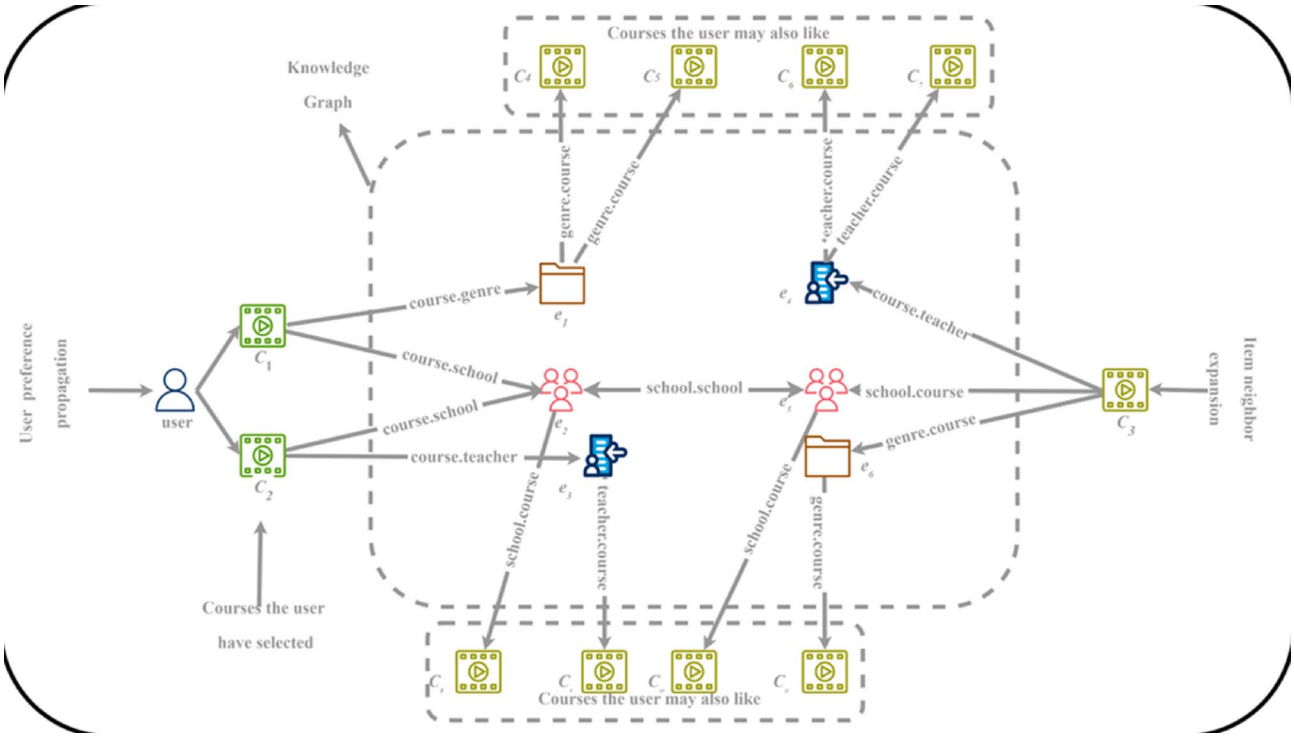

Fig. 2. The toy example of a KGCN-UP course recommendation for each user.

Related works

A variety of methods have been developed to introduce KG into recommendation systems. We categorize these methods into four types based on the different uses of additional information from the KG: Embedding-based[8–10], Path-based[14–16], GNN-based[18–22] and Contrastive Learning-based[27–29].

Embedding-based methods

Embedding-based methods[8–10] utilize knowledge graph embeddings (KGE)[11,12] to process KG information, aiming to incorporate both entity and relation embeddings into recommendation systems. CKE[8] uses the TransR[11] algorithm to extract the structured information of items, integrating it into the collaborative filtering module. DKN[9] aggregates target items by combining entity embeddings and word embeddings through the TransD[13] algorithm. RippleNet[10] propagates entities from previous user interactions within the KG, modeling the relationships between users and items. These models primarily focus on the semantic relevance between items, but they fall short in capturing complex high-order relationships between users and items, and are more commonly used in prediction tasks.

Path-based methods

Path-based methods[14–16] explore potential relationships between users and items in the KG by investigating the paths between them. PER[14] extracts all meta-paths from user-item data to explore the possibilities of user-item interactions. HIN[15] treats the KG as a heterogeneous information network, effectively utilizing its semantic information to represent users and items for personalized recommendations. KPRN[16] extracts paths between users and items from the KG and models entities and relationships by using LSTM[17] to infer user preferences. These models require rigorous domain knowledge and heavily rely on manually set meta-paths, making them a time-consuming and labor-intensive task.

GNN-based methods

Graph Neural Network (GNN) methods[18–22] establish mechanisms for information aggregation in GNNs, extracting multi-hop information from neighboring nodes and integrate it into node representations to learn user preferences and model long-range connections. Models such as KGCN[18] and KGNN-LS[19] utilize Graph Convolutional Networks (GCN) to synthesize neighbor information, capturing local neighbor structures to obtain item embeddings. KGAT[20] combines interaction graph and KG into heterogeneous graph, then recursively uses GCNs for information aggregation, thereby enhancing entity embeddings. CKAN[21] introduces a novel heterogeneous propagation mechanism to transmit collaborative signals and knowledge-aware information, thus dynamically adjusting the weights of neighboring nodes. Recently, KGIE[22] connects KG with user-item interaction matrices to form relational interactive embeddings for users, and then aggregates neighbor information through Convolutional Neural Networks (CNNs). Unlike these models, which make indirect use of relational information in the graph, our proposed method directly leverages the rich relational information

within the KG. It places greater emphasis on the types of relationships themselves, rather than just the entities connected by these relationships, resulting in improved recommendation performance.

Contarstive-learning based methods

Contrastive-learning based methods[27–29] involve comparing positive and negative samples, positioning the positive samples in the feature embedding space, while further separating the negative samples from the positive ones. The goal is to maximize the distance between different samples, improving node representation learning through supervised learning from the data itself. KGCL[27] aims to suppress the noise generated in the information aggregation process of KG by introducing SGL[30] into recommendation systems, utilizing self-supervised signals to enhance the representations of users and items. MCCLK[28] conducts contrastive learning (CL) from both local and global perspectives, enhancing node representations through multiple views. KGrec[29] calculates scores for each triplet, aligning signals from KGs and interaction graphs to mask the KG.

Methods

In this section, we define the recommendation task and introduce related concepts. Following that, we describe our proposed KGCN-UP model, which efficiently captures high-order interaction information between users and items, thereby improving the use of side information in the KG.

Problem definition

In this paper, we follow the general setup of mainstream recommendation systems. We define two types of structured data required by the course recommendation system: the user-course interaction dataset and the KG. Finally, we formulate the problem statement.

1) Interaction Data. Let $U=\{u_1, u_2, ...u_n\}$represent a set of m users, and let $C = \{c_1, c_2, ...c_n\}$ represent a set of n courses. Thus, the user-course interaction matrix can be defined as $Y = \{ y_{uc} | u \in U, c \in C\}$, where U and C denote the sets of users and courses. In this interaction matrix, if user u interacts with course c, such as by clicking, adding, or purchasing, then y_{uc}=1, otherwise y_{uc}=0.

2) Knowledge Graph. KGs include much side information. We formally define it as $G = \{(h, r, t)| h, t \in \epsilon, r \in R\}$ where h and r represent the head and the relation of the knowledge triplet, respectively, and t represents the tail. A KG is composed of many such knowledge triplets, which are the basic units of the graph. For example, the triplet (Nanjing University. Operating System, course. teacher, Luo Bin) describes the fact that the course on Computer Operating Systems is taught by Professor Luo Bin. To align more closely with real recommendation scenarios, We ensure that each course $c \in C$ corresponds to one entity $e \in \epsilon$ in the KG, establishing a set of item-entity alignment pairs $K = \{(c, e)| c \in C, e \in \epsilon\}$.This alignment between courses and entities in the KG provides rich semantic information for interaction data.

3) Problem Statement. The primary goal of the online course recommendation problem is to predict whether a user u is interested in a course c that they have not yet interacted with, using the interaction matrix Y and the knowledge graph G. The main task of the model is to construct a prediction function $y_{uc} = F(u, c|G, \Theta)$, where y_{uc} represents the predicted likelihood of student u engaging with course c, F denotes the functional form of the model, and Θ represents the parameter of the function F.

Framework

As illustrated in Fig. 3, the architecture of the KGCN-UP model is structured into three primary components: the User Preference Propagation Module, the Item Neighbor Enhancement Module, and the Prediction Module. The User Preference Propagation Module explores the user's potential preferences through various relations in the KG, enriching the user's preference representation. The Item Neighbor Enhancement Module generates the

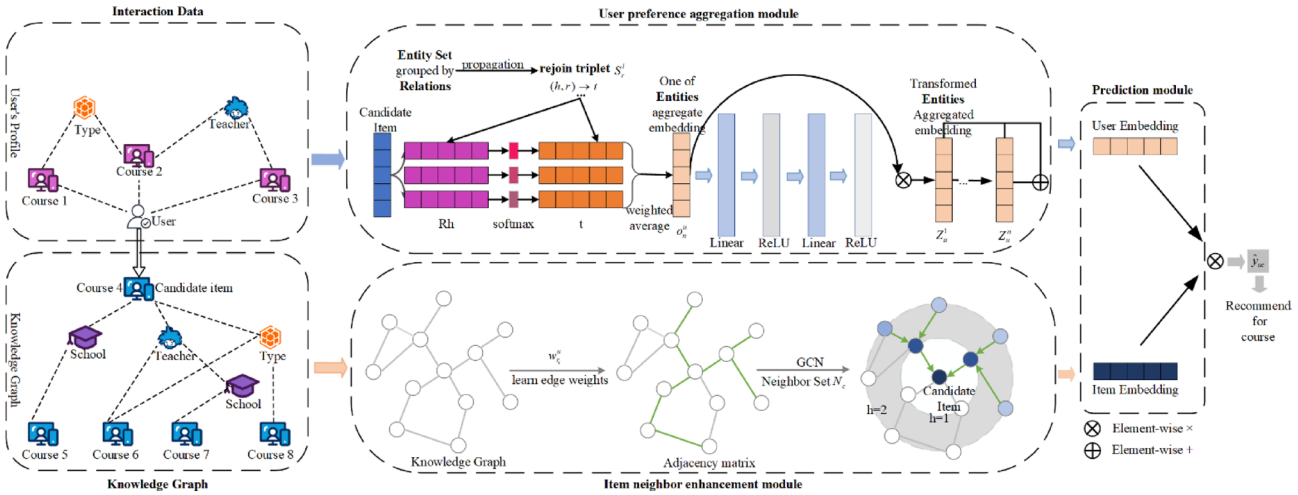

Fig. 3. The proposed KGCN-UP model is illustrated, consisting of three modules: the user preference propagation module, the item neighbor enhancement module, and the prediction module.

final candidate course list by capturing higher-order neighboring items of the target course and aggregating similarities and correlations among these items. Finally, the Prediction Module integrates the merged user and item representations and outputs the prediction of the user's selection of candidate courses.

User preference propagation module

Similar to the assumptions of traditional collaborative filtering (CF) models[31–33], we assume that users' preferences are mainly reflected in their historical interaction data. Specifically, users tend to choose courses that are similar to items they have engaged with in the past. However, this assumption encounters numerous challenges in practical applications, especially in MOOC scenarios. The number of users usually far exceeds the number of courses, and user behavior often concentrates on popular courses. This leads to extremely sparse historical interaction data, making it difficult to utilize user information to identify potential key users.

The entities in KG contain fruitful facts and connect. For instance, the course ''University Physics I" is associated with ''Tsinghua University" (school), the ''Maxwell Rate Distribution Law" (concept), ''Chen Xinyi" (teacher), and ''Physics" (category). However, ''Chen Xinyi" has a deeper connection with the courses he teaches, such as ''University Physics II" and ''University Physics" (Prerequisite). If a user has previously watched Mr. Chen's courses, he is likely to be interested in physics or other courses taught by Mr. Chen, and may click on ''College Physics II (Mechanics, Heat)" or ''College Physics" (Prerequisite) for further learning. Therefore, a user's personality and preferences should be defined not only by the courses they have previously watched but also by other relevant entities in the KG. To improve recommendation performance, we should enhance user representation to get user-oriented information through user preference sets and KGs. The detailed process is as follows:

User preference propagation set

We consider the set of entities that the user has interacted with as the initial seed in the KG. The initial set is defined as follows:

$$\epsilon_u^0 = \{e|\,(e\,,\,c) \in\ K\ and\ c \in\ \{c|y_{uc} = 1\}\} \tag{1}$$

The initial set consists of course entities, with each course entity serving as the head entity. This head entity is explored along the various relationship chains in the KG through ϵ_u^0 to get the tail entities. These tail entities then serve as new head entities in subsequent rounds. This process iterates l times to obtain the $l-hop$ propagation entity set. This set can be seen as a natural expansion of the user's interests. The user preference propagation set is defined as follows:

$$P_u^l = \{t|\,(h\,,\,r\,,\,t) \in\ G\ and\ h \in\ P_u^{l-1}\},\ l = 1,2,...,L \tag{2}$$

The initial set p_u^0 represents the courses chosen by the user, with the number of courses being equal to ϵ_u^0. To facilitate the calculation of similarity between courses previously selected by the user and candidate courses, we define a set of triples S_u^l as follows:

$$S_u^l = \left\{(h\,,\,r\,,\,t)\,|\,(h\,,\,r\,,\,t) \in\ G\ and\ h \in\ P_u^{l-1}\right\},\ l = 1,2,\ldots,L \tag{3}$$

S_u^l differs from p_u^l in that each element of serves as the head entity in the set of S_u^l triples. The set S_u^l explore additional related items by traversing various relationship chains in the KG to find other related items, this process is crucial for MOOC recommendations.

Path grouping propagation set

Although the rich semantic information of the KG expands user representation, having either too many or too few elements in each layer set S_u^l will degrade the recommendation effect. For example, the course "Data Structures" includes 140 concepts, which constitutes a one-to-many relationship. In contrast, the relationships between the course and the school, and between the course and the teacher, are one-to-one. Mixing these knowledge points into the S_u^l set will make it difficult to identify which knowledge points in "Data Structures" user is interested in and also distract the user's attention. Having too few elements in each layer results in an insufficient number of knowledge entities per connection, which not only reduces the effectiveness of enriching the user's encoding but also risks failing to iterate to the $l-th$ level. Since different relationship chains in the KG can be seen as extensions of various attributes of the same course by the user, they naturally have the advantage of grouping. Based on this, we obtain the path grouping propagation set by traversing through P_u^l the relationship chain, as follows:

$$P_r^l = t|\,(h,r,t) \in\ G, h \in\ \ P_u^{l-1} and r \in\ R, l = 1,2,\ldots,L \tag{4}$$

Where r is the link which belongs to in the entity alignment set K. Similarly, for ease of calculation, the user's path grouping propagation set is extended into a triplet entity set, as follows:

$$S_r^l = \{(\,h\,,\,r\,,\,t\,)|\,(h\,,\,r\,,\,t) \in\ \ G\ and\ h \in\ \ P_r^{l-1}\},\ r \in\ R \tag{5}$$

Path preference aggregation set

For a given user u and a candidate course c, tail entities can be expressed by different head entities via various relationship chains. For instance, the course ''Bioinformatics" may be offered by ''Central South University" or taught by ''Tong Jianbin". Therefore, the semantics of tail entities, derived from different head entities and relationship chains, vary across different triples. To more finely represent the relationship between the varying semantics of tail entities and the candidate course c, it is necessary to perform a weighted representation of the resulting tail entities within the given grouping set S_r^l and across different triples. This approach aims to calculate the similarity of candidate courses to the courses that users interact with through different relationship chains.

$$a_i = softmax\left(c^T R_i h_i\right) = \frac{exp\left(c^T R_i h_i\right)}{\sum_{(h,r,t)\in S_r^l} exp\left(c^T Rh\right)} \tag{6}$$

Where $R \in \Re^{d\times d}$ denotes the vector representation of relationship r_i, and d is the dimension of the embedding. Similarly, $h \in \Re^d$ represents the embedding of the head entity in S_r^l while c^T is the transpose of the course embedding vector c. The resulting similarity can be interpreted as the probability that the grouping set S_r^l is associated with a particular triple (h_i, r_i, t_i). We then weight and sum all the tail entities in each hop to get a weighted representation of the different paths:

$$O_n = \sum_{(h_i,r,t_i)\in S_r^1} a_i t_i \tag{7}$$

where $t_i \in \Re^d, n = |R|$.

The different sets of path grouping in each propagation layer can be regarded as an extension of users' interests in a certain direction. To reveal the users' interest hotspots, it is necessary to determine which types of course information attract more user attention by using attention weights. The attention function is defined as follows:

$$z_n = \zeta\ (o_n, r_n)\, o_n \tag{8}$$

The function $\zeta\ (o_n, r_n)$ determines the weight of each path, reflecting the user's preference for different relation-links. Inspired by Vaswani's attention mechanism[34], $\widetilde{\zeta}\ (o_n, r_n)$ is defined as follows:

$$\widetilde{\zeta}\ (o_n, r_n) = ReLU\left(W_2(W_1(o_n||r_n) + b_1) + b_2\right) \tag{9}$$

Here, $ReLU$ serves as a nonlinear activation function, while W and b are trainable parameters representing linear transformation matrices and bias terms, respectively. The operator || denotes the concatenation of two vectors. The subscripts on W and b indicate that these parameters belong to different layers. By processing in this way, scores for different paths chosen by the user are obtained. Subsequently, to convert attention scores into weights, it is necessary to normalize the coefficients of the entire relationship path. Subsequently, to convert attention scores into weights, it is necessary to normalize the coefficients of the entire[36]:

$$\zeta\ (o_n, r_n) = \frac{exp\left(\widetilde{\zeta}\,(o_n, r_n)\right)}{\sum_{i\in|R|} exp\left(\widetilde{\zeta}\,(o_n, r_n)\right)} \tag{10}$$

Therefore, we have determined which type of course knowledge the learners focus on more.

Finally, the attention weights obtained are added to the propagation triples of different paths, resulting in a user embedding representation based on the user preference propagation module as follows:

$$e^u = \sum_{i=1}^{l} \sum_{n=1}^{|R|} z_n^u \tag{11}$$

Item neighbor enhancement module

In the recommendation system, we posit that the final representation of candidate items is influenced not only by their own representation but also by the attributes of their neighboring entities in the KG. Specifically, the process begins with the candidate item, from which we construct a set of related entities by iteratively traversing different relational chains within the KG. Subsequently, by calculating the attention weights based on the information in the entity set, we implement message passing based on the attention mechanism, thereby capturing the neighbor information adjacent to the candidate item. This information is subsequently aggregated at the central node to form the final representation of the item. This approach not only captures the potential relationships between different entities but also aggregates neighborhood information of the candidate items, thereby elucidating how relational preferences impact user behavior. The detailed process is as follows:

Obtain first-order neighborhood

By traversing the KG to collect a set N_c of one or more entities directly related to the candidate course $c \in \Re^d$, we refer to it as the first-order neighbor set of course c. In the process of message aggregation, using the relation vector R_i directly as the weight for message passing would obscure the different impacts of relationships on user behavior. For example, consider students who have opted for the course "C++Programming Design". Some chose it because they needed a foundation in "C++Programming Design" for "Data Structures"; some because

of the instructor Zheng Li; and others chose it because the course is offered by Tsinghua University. Therefore, we represent each edge with a weight w:

$$w_{r_i}^u = q\left(e^u, R_i\right) \tag{12}$$

In this equation, $e^u \in \mathfrak{R}^d$ represents the vector of the user, and $R_i \in \mathfrak{R}^d$ is the relation vector where e^u is connected to the ith neighbor. The function q calculates the correlation score between the user vector and the relation vector through an inner product, resulting $w_{r_i}^u$. This score $w_{r_i}^u$ expresses the target user's degree of preference for the relation R_i, representing the weight of message passing through the edge R_i. Subsequently, the obtained $w_{r_i}^u$ is normalized:

$$\widetilde{w}_{r_i}^u = Softmax_j\left(w_{r_i}^u\right) = \frac{exp\left(w_{r_i}^u\right)}{\sum_{j \in N_c} exp\left(w_{r_j}^u\right)} \tag{13}$$

where N_c represents the set of first-order neighbors of the candidate course c.

The final embedding of the candidate course c, which incorporates user representation, is obtained by weighting the adjacent entities associated with the course:

$$\widetilde{e}_{N_c}^c = \sum_{i \in N_c} \widetilde{w}_{r_i}^u \cdot e_i \tag{14}$$

where $e_i \in \epsilon$, represents the vector of the $i\text{-}th$ neighbor.

As shown in Fig. 3, inspired by graph sampling[37,45], the message passing process spreads from the outermost to the innermost layer. Through aggregation at each layer, the inner layers near the item c aggregates features from outer item entities, ultimately converging at the central node. This process yields the feature vector $\widetilde{e}_{N_c}^c$ for the final candidate course c, capturing the higher-order semantic information from the KG.

Iteration layer

On one hand, it is considered that the actual sizes of N_c in real-world KGs vary; on the other hand, if a candidate item aggregates too many neighbors, it will significantly increase the computational load on the model. Considering these factors, we fixed the number of first-order neighbors sampled randomly for each candidate course. The resulting set of neighbor entities denoted as $e_{S_c}^c$, do not include all neighboring entities, thus enabling more efficient and manageable computations. Specifically, the subset $e_{S_c}^c$ for a candidate can be defined as $S_c \rightarrow \{e|\, e \sim N_c\}$, where $|S_c| = D$ and D is a configurable constant, set to 2.

The candidate course vector e^c and the neighborhood set vector $e_{S_c}^c$ are combined into a single vector using a summing aggregator function $f_{agg} : \mathfrak{R}^d \times \mathfrak{R}^d \rightarrow \mathfrak{R}^d$. The aggregation process formula is as follows:

$$f_{agg}\left(e_{S_c}^c, e^c\right) = e_{S_c}^c + e^c \tag{15}$$

$$e^c = \sigma\left(W \cdot f_{agg}\left(e_{S_c}^c, e^c\right) + b\right) \tag{16}$$

The summation aggregator combines two vectors by addition, where σ () represents a nonlinear activation function, W is the linear transformation matrix, and b is the bias term.

With the addition of iterative layers, the model can progressively extract information across multiple hops and integrate it into the course, obtaining the final embedding representation through the item neighbor enhancement module.

Prediction module

As depicted in the Fig. 3, the prediction module processes the resulting user embeddings and candidate course embeddings as input vectors to output the prediction results. The specific steps are as follows:

Given the user embedding e^u and the course embedding e^c, we calculate the probability of the user selecting the course by computing the inner product of e^u and e^c, as follows:

$$\widehat{y}_{uc} = sigmod\left(e^{u^T}, e^c\right) = \frac{e^{u^T} e^c}{1 + exp(-x)} \tag{17}$$

To balance positive and negative samples and enhance classification performance, negative sampling is typically used to randomly generate an equal number of negative samples for each user. Cross entropy is then employed to quantify the discrepancy between the predicted outcomes and the actual interactions. The loss function is defined as follows:

$$\mathcal{L}_{CF} = \sum_{u \in U}\left(\sum_{c \in c|(u,c) \in \rho^+} \iota\ (y_{u,c}, \widehat{y}_{u,c}) - \sum_{c \in c|(u,c) \in \rho^-} \iota\ (y_{u,c}, \widehat{y}_{u,c})\right) \tag{18}$$

Where the function $\iota\ (y_{u,c}, \widehat{y}_{u,c})$ represents the cross-entropy loss. ρ^+ denotes the interactions in which the user actually selects or shows interest, ρ^- represents the set of all negative user-course interactions, and ρ is assumed to follow a uniform distribution.

To capture the potential representation of the relationships between entities as possible, we adopt the approach[10] and utilize KGE method to constrain the embeddings of entities and relationships. The loss function for the KG is defined as follows:

$$\mathcal{L}_{KG} = \sum_{(h,r,t)\in G} || I_r - E^T R E||_2^2 \tag{19}$$

Where I_r represents the slice of the tensor I corresponding to relation r in the KG, and E is the embedding matrix of entities.

Finally, to mitigate model overfitting and reduce noise interference, we introduce a regularization loss function:

$$\mathcal{L}_{reg} = \sum_{r\in R} || r||_2^2 + \sum_{c\in \epsilon} || c||_2^2 + \sum_{e\in \epsilon} || e||_2^2 \tag{20}$$

Therefore, the loss function of the model can finally be expressed as:

$$L = \mathcal{L}_{cf} + \lambda_1 \mathcal{L}_{kg} + \lambda_2 \mathcal{L}_{reg} \tag{21}$$

Where λ_1 and λ_2 are hyperparameters that help prevent overfitting.

Input: The interaction data Y; the knowledge graph G.
Output: The predicted value $y_{uc} = F(u, c|G, \Theta)$ that user u will engage with item c.
1: Initializing the parameters Θ
2: $S_r^l \leftarrow$ Generate an aggregated set of triples for each user;
3: $e^u \leftarrow$ Calculate the user embedding in the S_r^l set and aggregate with the candidate item
4: $e^c \leftarrow$ Combine neighbor information and aggregate multiorder neighborhoods
5: for each training epoch ranging from 1 to 512 do
6: Sample the smallest batch of positive and negative samples from the Y.
7: Sample the smallest batch of positive and negative triples from the G.
8: Update the parameters Θ in F using gradient descent based on the learning rate η.
9: end for
10: return $F(u, c|G, \Theta)$

Algorithm 1 KGCN-UP

Experiment

In this section, we conducted a series of experiments to verify the effectiveness of the proposed KGCN-UP model in online course recommendations. We aimed to answer the following research questions:

RQ1: How does our proposed model perform compared to other advanced models that also aim to improve recommendation performance?

RQ2: How do hyperparameters impact model performance?

RQ3: How do the user propagation modules and item enhancement modules influence the performance of the model?

RQ4: Can KGCN-UP offer visual case studies to demonstrate the effectiveness of the proposed appr

Datasets prepare

To verify the effectiveness of the proposed model in online course recommendations, we conducted experiments using the MOOCCube open dataset. Additionally, to ensure the fairness of the results, we also performed comparative experiments on two other public datasets, Movielens-1 M and Book-Crossing, to assess the model's universality in the recommendation scenario.

1)MOOCCube: It is a large-scale and open-source educational dataset that not only includes 706 real courses and nearly 38,000 videos, but also provides additional information for course recommendations.

2)MovieLens-1 M: This dataset, widely used for movie recommendations, includes display ratings from 6,036 users for 2,445 movies, with ratings on a scale of 1 to 5.

3)Book-Crossing: It collects reviews of 2,445 books from 17,860 readers across communities (on a scale of 0 to 1).

We preprocessed the MOOCCube open dataset with specific steps to enhance data quality and relevance. Initially, we filtered the interactive data collected between July 1, 2017, and October 1, 2017. We removed records where learners selected fewer than 10 courses to reduce data sparsity and ensure a sufficient amount of interactive information. Subsequently, we extracted all course names from the dataset, maintaining the historical sequence of learners' course clicks. Regarding the construction of the sub-KG, we start from the course entity

Datasets		Book-Crossing	MovieLens-1 M	MOOCCube
User-Item Interaction	#user	17,860	6036	2396
	#item	14,967	2445	592
	# interactions	139,746	753,772	32,091
Knowledge Graph	#entities	77,903	182,011	2092
	#relations	25	12	7
	#triples	151,500	1,241,996	20,893

Table 1. Statistics settings for the datasets.

Hyper-parameter settings	MOOCCube	Book-Crossing	MovieLens-1 M
h (hop numbers)	2	3	2
n (neighbor sampling size)	8	8	8
s (the size of each triplet)	32	64	32
λ_1 (kge weight)	10^{-2}	10^{-2}	10^{-2}
λ_2 (regularizer weight)	10^{-4}	2×10^{-5}	10^{-4}
η (learning rate)	10^{-4}	10^{-4}	10^{-4}

Table 2. Statistics hyper-parameter settings for the datasets.

and expand into various course-related knowledge triples through different relationships, until the tail entity points back to the course entity.

For the interaction data from the MovieLens-1 M and Book-Crossing datasets, which exhibit explicit feedback, we adopted the RippleNet[10] approach to convert this data into implicit feedback, thus capturing user-item interactions. For MovieLens-1 M, we set a positive sample threshold of 4; for Book-Crossing, we did not set a specific threshold due to its sparse data. A user's engagement with an item is quantified such that a positive interaction is assigned a score of 1, whereas the absence of interaction or a low rating is assigned a score of 0. Negative samples are randomly selected from unobserved items for each user, ensuring the number of negative samples equals the number of positive samples. In terms of sub-KG construction, we followed the KGCN[18] methodology and utilized Microsoft's Satori[4] business KG to construct the sub-KG for the MovieLens-1 M and Book-Crossing datasets. To create the sub-KG, we first filter out low-confidence triples from the original knowledge graph (KG) by selecting only those with a confidence level greater than 0.9. This pruning step helps eliminate noisy or unreliable data. For the remaining triples, we extract the Satori IDs of all valid movies and books by matching their names with the tail of the triples, such as (head, film.film.name, tail) and (head, book.book.title, tail). To further reduce noise, we exclude items that either match multiple entities or have no valid match at all, ensuring that only clear and unambiguous entities are retained. Table 1 presents the statistics for these three datasets.

Experimental environment and training settings

To ensure the fairness of the experiments, we standardized the model's dimension at 64 and the batch size at 512. The dataset was randomly divided into training, testing, and validation sets in a 6:2:2 ratio to facilitate hyperparameter tuning. The model parameters were initialized using the Xavier[38] method, and the model was optimized with the ADAM optimizer[39]. The hyperparameters that need to be fine-tuned are as follows: evaluate learning rates among {0.0001, 0.001, 0.01, 0.1}, check regularization coefficients among {10^{-7},10^{-6},10^{-5},10^{-4},10^{-3},10^{-2},10^{-1}} explore the maximum number of hops L among {1, 2, 3, 4, 5}, and find the appropriate number of neighbors D among {2, 3, 4, 5, 6, 7, 8}. Training was halted if the model's loss on the validation set did not improve significantly over five consecutive validation cycles. The final results were averaged from the best outcomes across five model runs. Table 2 lists the specific hyperparameter settings for our model, while the hyperparameters for other baseline models are set based on experience or in accordance with the settings in the original papers to ensure optimal performance. The experiments were conducted on hardware equipped with NVIDIA GeForce RTX 4050 GPUs, with all models implemented in Pytorch and sharing the same hardware for acceleration.

Evaluation metrics

To evaluate the performance of the proposed method, we predict user clicks by using the trained model and assess the effectiveness of the click-through rate (CTR) prediction by employing two common evaluation metrics: AUC (Area Under Curve) and F1 score. AUC is an evaluation index that measures the performance of a recommendation system on a scale from 0 to 1, with higher values indicating better performance. The F1 score is an indicator that balances precision and recall, also ranging from 0 to 1, where a higher value indicates better performance.

Baselines

To validate the performance of the proposed model KGCN-UP, we compared it with five classical recommendation approaches categorized into different methods: CF-based (LFM, BPRMF), Embedding-based (CKE, MKR, RippleNet), Path-based (PER), GNN-based (KGCN, KGNN-LS, KGAT, CKAN, KGIN, KGIE, CG-KGR), and CL-based (KGCCL, KGRec). Each category is described as follows:

1) LFM[40]: A collaborative filtering (CF) method that utilizes Alternating Least Squares (ALS) decomposition to derive implicit factors for users and items, constructing an interaction matrix that represents user ratings of items.

2) BPRMF[41]: A CF approach that employs pairwise matrix decomposition to capture implicit feedback and ranks user preferences using Bayesian Personalized Ranking (BPR) loss optimization.

3) CKE[41]: An embedding-based approach that integrates KG into the recommender system using the TransR [] algorithm to extract structured knowledge, which is then fused with CF techniques.

4) MKR[42]: An embedding-based method that performs recommendation prediction tasks concurrently with knowledge graph embedding (KGE), using cross compression units that are updated synchronously to enhance recommendation outcomes.

5) RippleNet[10]: A path-based approach that propagates potential user preferences through various relational links in the KG, culminating in an item representation that incorporates user preferences.

6) PER[14]: A path-based method that represents user-item connections by calculating path similarity between items to derive preference scores, forming a user preference diffusion matrix.

7) KGCN[18]: A graph neural network (GNN)-based approach that enriches item representations by capturing higher-order semantic information in the KG through recursive message aggregation.

8) KGAT[20]: The embeddings of neighbor nodes are recursively updated while adopting an attention mechanism to dynamically adjust the weights of each neighbor, thereby distinguishing the importance of different neighbors in node embeddings.

9) CKAN[21]: A GNN-based approach that introduces a novel dissemination strategy combining collaborative and knowledge information, dynamically adjusting the weight of each neighbor node through an attention mechanism to enhance recommendations.

10) KGIN[43]: A GNN-based method that extracts rich semantic information from the KG by constructing relational path perceptions, refining key user intent features, and enriching the embedded representations of users and items.

11) KGIE[22]: A GNN-based method that constructs a user-item interaction matrix and fuses it with item information through GNNs, effectively capturing user information and preferences and considering user context in the recommendation process.

12) KGCCL[44]: A CL-based method that combines interaction graphs and KGs. By contrasting local and global level, it uses noise enhancement to improve the representations of users and items.

13) KGRec[29]: A CL-based method that calculates high-scoring triplets and then masks the KG by aligning signals from both the KG and the interaction graph.

Preformance comparison (RQ1)

The performance results of all models are shown in Table 3. By analyzing their performance differences, the following observations are obtained:

1) KGCN-UP achieved the best results. Specifically, compared to the strongest benchmark model, KGCN-UP shows significant improvements in AUC on the Book, Movie, and Course datasets respectively by 0.14%, 0.24%, and 4.94%. We attribute these improvements to: (1) the user preference propagation module, which better

	Book-Crossing		MovieLens-1 M		MOOCCube	
Model	AUC	F1	AUC	F1	AUC	F1
LFM	0.6063	0.5112	0.8354	0.7811	0.7232	0.6262
BPRMF	0.6578	0.6107	0. 8521	0.7763	0.7548	0.6678
CKE	0.6654	0.6112	0.9097	0.8024	0.7632	0.6994
MKR	0.7318	0.6548	0.9147	0.8432	0.7731	0.6998
RippleNet	0.7233	0.6554	0.9193	0.8441	0.8017	0.7223
PER	0.6042	0.5663	0.6224	0.5867	0.6174	0.5833
KGCN	0.6743	0.6213	0.8993	0.8261	0.7764	0.7028
KGAT	0.7312	0.6557	0.9132	0.8378	0.7914	0.7154
CKAN	0.7397	0.6642	0.9047	0.8363	0.8097	0.7315
KGIN	0.7263	0.6582	0.9113	0.8424	0.7923	0.7214
KGIE	0.7269	0.6584	0.9214	0.8538	0.8143	0.7443
KGRec	0.7231	0.6658	0.9219	0.8547	0.8176	0.7552
KGCCL	0.7271	0.6593	0.9205	0.8534	0.8144	0.7449
KGCN-UP	0.7408	0.6699	0.9342	0.8554	0.8580	0.7719

Table 3. Performance comparison on different datasets.

captures the higher-order preferences of users; (2) the item neighbor enhancement module, which effectively utilizes the higher-order semantic information from the KG.

2) The introduction of KG information not always enhances the recommender system. Compared to traditional recommendation algorithms like LFM and BPRMF, the CKE model, which embeds knowledge information into matrix factorization, shows a notable performance improvement. This indicates the effectiveness of incorporating knowledge information, aligning with conclusions from previous studies[7].While PER performs worse than BPRMF, this indicates that the model's performance can only improve by incorporating appropriate knowledge. This highlights the critical role of knowledge sampling and knowledge denoising. This proves that the method we proposed can effectively handle the application of knowledge graph auxiliary information without turning it into a burden that leads to performance degradation.

3) GNN-based methods exhibit stronger performance. Most of the GNN-based methods outperform embedding-based and path-based methods. This highlights the importance of information propagation between graph nodes and suggests that enhancing the representation of users and items can improve model performance, especially in scenarios with sparser interaction data. This also proves that propagating user preference information can achieve good performance in the sparse MOOC scenario.

Parameter analysis (RQ2)

Our proposed KGCN-UP model involves several hyperparameters, as detailed in Fig. 4. We will specifically discuss how these parameters affect the model's performance.

1) Impact of Embedding Dimension (d). We investigated the impact of the embedding dimension on the performance of the KGCN-UP model. As shown in Fig. 4, model performance initially tends to increase with an increase in d. This improvement is mainly because the model can encode more information from users and entities. However, beyond a certain point, an increase in d starts to decrease performance, likely due to higher dimensions introducing excessive noise, which leads to model overfitting and adversely affects performance.

2) Impact of Number of Neighbors (n). We investigated the impact on the performance of the KGCN-UP model by adjusting the size of neighbors. As shown in Fig. 4, the optimal performance of the KGCN-UP model is achieved when $n = 8$. This may be because if n is too small, the model cannot acquire enough neighbor information, while a large n may introduce excessive noise, leading to interference in the feature representation of the target entity. Therefore, finding an appropriate number of neighbors is crucial for the effectiveness of the model.

3) Impact of Triplet Size (s). We further investigated the robustness of the model by varying the size of each triplet in the path grouping propagation ensemble. As shown in Fig. 4, an initial increase in s significantly improves model performance, as it allows the model to encode more knowledge from the KG. However, extracting too much information can cause the number of parameters to skyrocket, increasing the training time and potentially reducing model performance when s is too large.

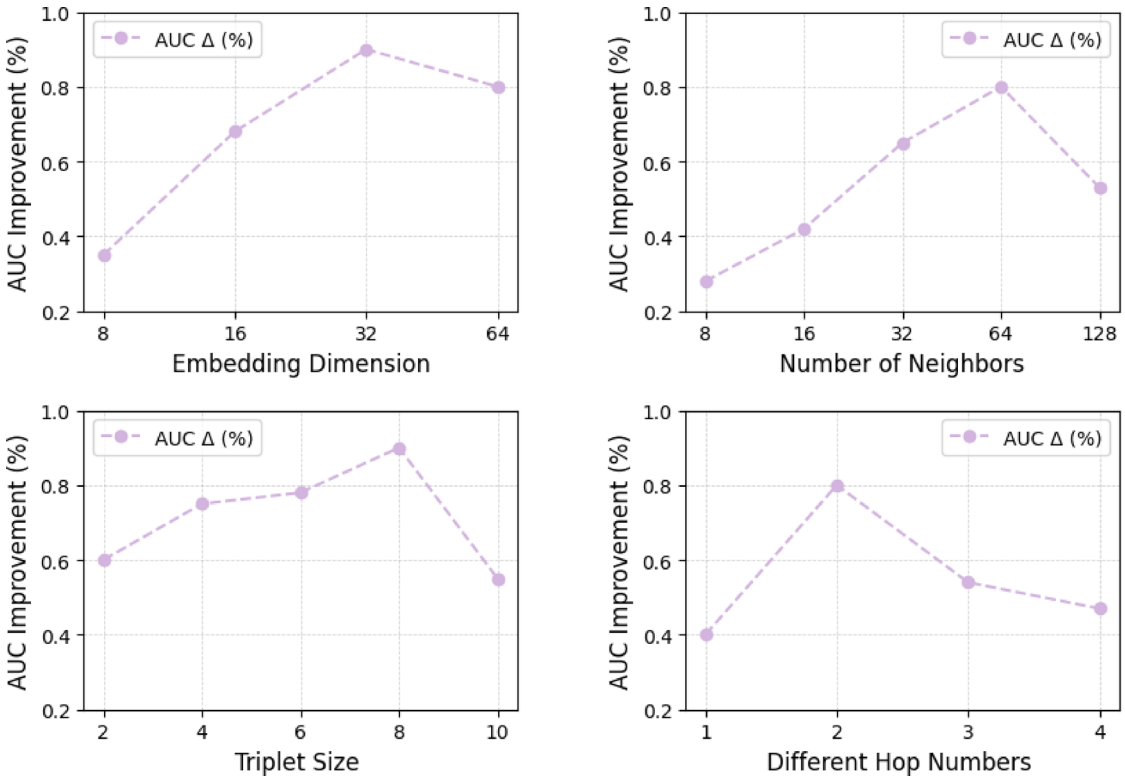

Fig. 4. Parameter sensitivity of KGCN-UP.

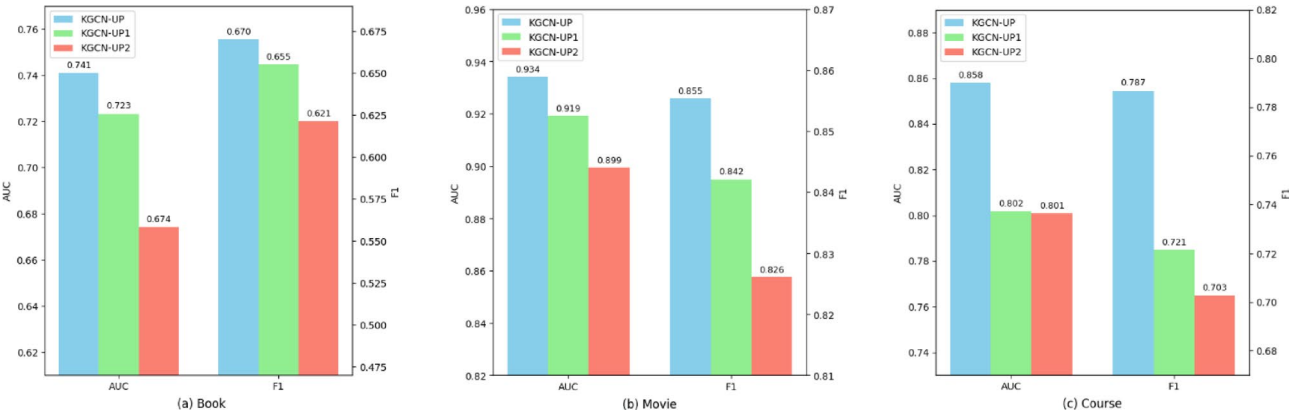

Fig. 5. Effect of ablation study.

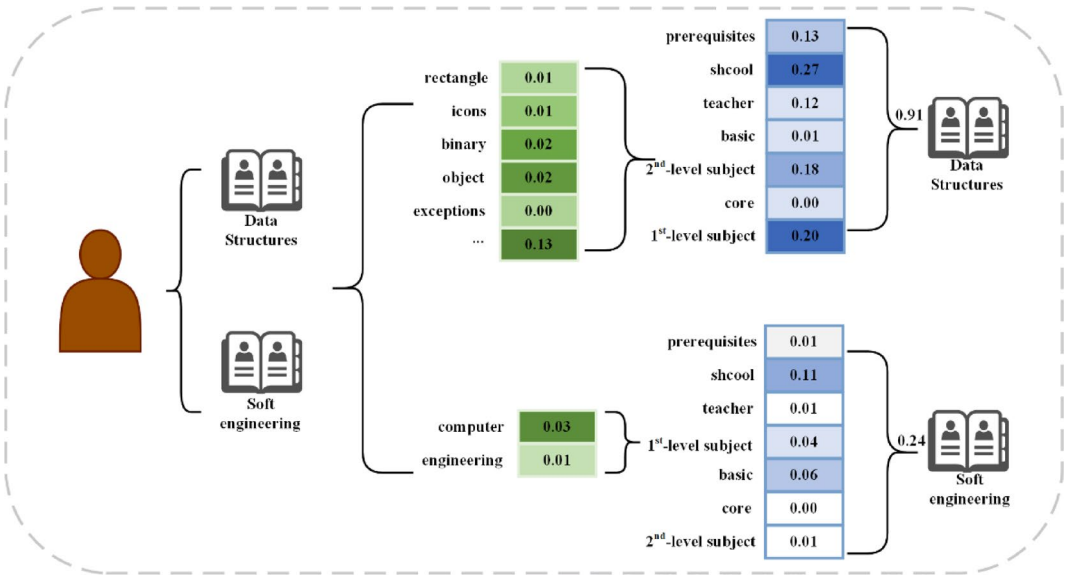

Fig. 6. The case study of KGCN-UP. The blue labels indicate the types of entities the user is interested in, while the green labels further refine the focus of attention.

4) Impact of Different Hop Numbers (h). We modified the numbers of hop in our experiments to investigate its impact on model performance. The results, as shown in Fig. 4, indicate that the optimal performance is achieved when h is set to 2. Analysis of this phenomenon suggests that a larger h enhances the model's ability to explore information from distant nodes within the KG, but it also introduces a significant amount of noise. This noise can overwhelm valuable signals, thereby negatively affecting the model's performance. Conversely, setting h too small limits the model's ability to utilize the relevancy between nodes in the graph, which similarly impairs performance.

Ablation comparison (RQ3)

To verify the contribution of the main components in the model to the overall performance, we compared KGCN-UP with the following two variants:

- KGCN-UP1: A variant of KGCN-UP that removes the user preference propagation module.
- KGCN-UP2: A variant of KGCN-UP that removes the item neighbor enhancement module.

Figure 5 shows the results for the two variants and KGCN-UP, from which we observed the following:

1) Removing the user preference module significantly decreases model performance, confirming the effectiveness of exploring latent user preferences through traversing the KG.

2) Removing the item neighbor enhancement module also leads to a performance decline, highlighting the importance of capturing high-order semantic information in the KG.

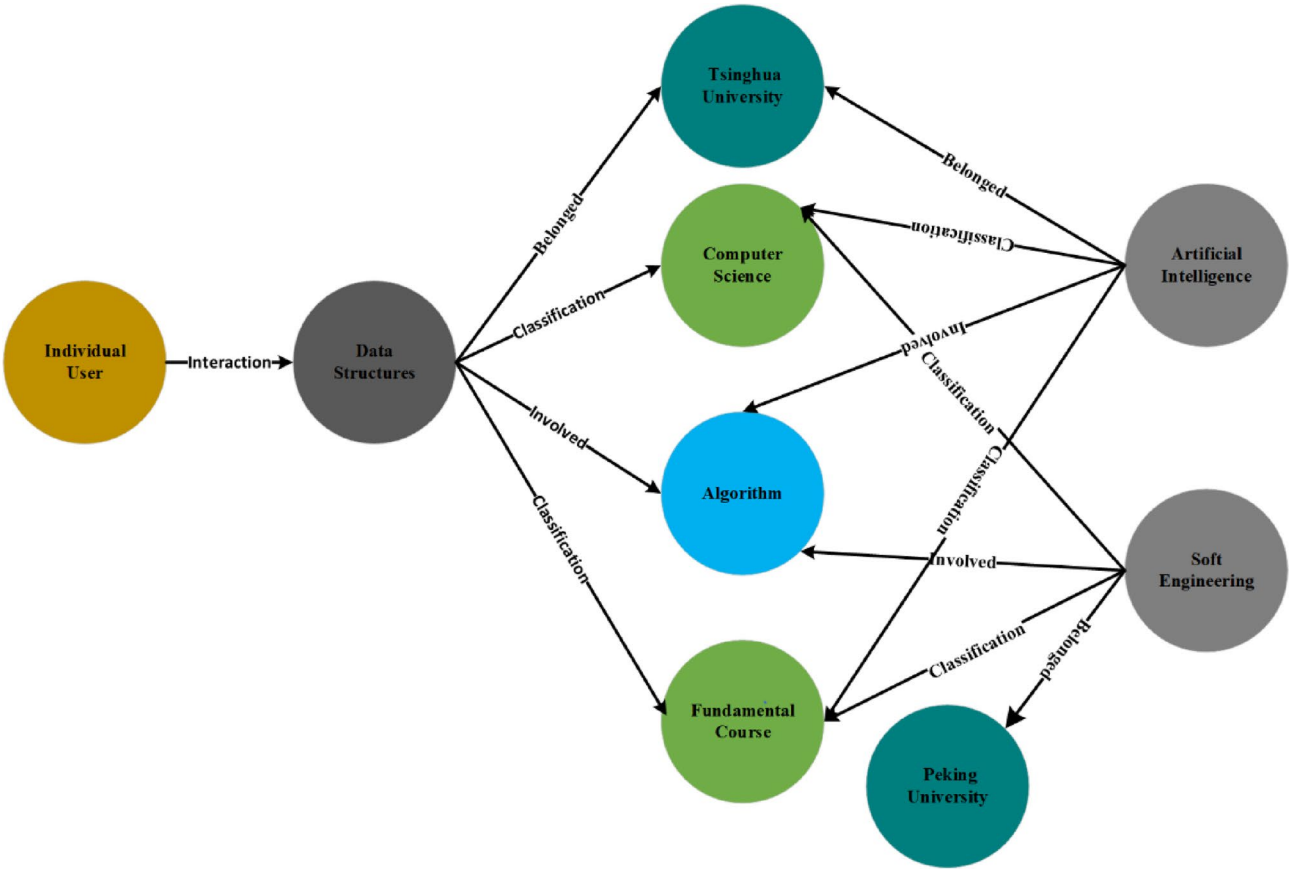

Fig. 7. The attention-based preference selection of the KGCN-UP model.

Case study (RQ4)

In this section, we demonstrate the effectiveness of the proposed KGCN-UP in a CTR prediction task, as shown in Fig. 6. From the visualization, it's very clear that the proposed model generates different prediction results for various candidate courses. The user's preferences are obvious: they have a greater interest in courses related to schools and prefer courses with content that aligns with their interests. For instance, although Software Engineering is also offered by Tsinghua University, the model predicts a low probability of this user selecting it. This is because the content of Software Engineering is less related to Data Structure, which is the user's main area of interest. On the other hand, courses that are more closely related to computer science, like Data Structure, show a higher chance of being chosen.

As shown in Fig. 7, the learner selects the Data Structures course on the MOOC platform. Although both Artificial Intelligence and Software Engineering belong to the computer science field and involve algorithm knowledge, the system ultimately recommends the Artificial Intelligence course to the user. This is because the KGCN-UP model, through the extraction of user preferences, identifies that the user prefers courses offered by Tsinghua University. This helps in understanding the patterns in how users select courses and provides a more intuitive visualization of their decision-making process. As a result, the system recommends the Artificial Intelligence course, which is also offered by Tsinghua University.

Conclusion

In this paper, we propose an end-to-end recommendation system model that effectively integrates with KG. This model not only extends user preferences within the KG through a user preference propagation module, reducing the problem of data sparsity, but also captures high-order structural correlations between entities through a item neighbor enhancement module, enabling it to capture and utilize more complex structural information. Based on these features, the model can balance and optimize user and item representations by using two different structures. Our proposed KGCN-UP model demonstrates astonishing advantages compared to the current state-of-the-art baselines. For future work, we need to introduce Federated learning to store interaction data between users and items in a protected manner, avoiding exposure of private information to local clients.

Data availability

The datasets utilized in this research, such as MOOCCube, MovieLens-1 M, and Book-Crossing, are publicly accessible. They can be obtained via the links provided below. Additionally, all other data, including images and codes used in this study, are available upon reasonable request by contacting the corresponding author. MOOC-

Cube: http://moocdata.cn/data/MOOCCube. MovieLens-1 M: https://github.com/CCIIPLab/MCCLK/tree/main/data/book. Book-Crossing: https://grouplens.org/datasets/movielens/1m/

Received: 5 May 2025; Accepted: 29 July 2025
Published online: 18 August 2025

References

1. Fan, J. et al. Interpretable MOOC recommendation: a multi-attention network for personalized learning behavior analysis.. *Internet Res.* **32** (2), 588–605 (2022).
2. Lin, Y. et al. Adaptive course recommendation in MOOCs.. *Knowl. Based Syst.* **224**, 107085 (2021).
3. Jamali, M. & Ester, M. A matrix factorization technique with trust propagation for recommendation in social networks.Proceedings of the fourth ACM conference on Recommender systems. : 135–142. (2010).
4. Cheng, Y. et al. An image dataset for benchmarking recommender systems with raw pixels.Proceedings of the 2024 SIAM International Conference on Data Mining (SDM). Society for Industrial and Applied Mathematics, : 418–426. (2024).
5. Sun, Y. et al. Collaborative intent prediction with real-time contextual data.. *ACM Trans. Inform. Syst. (TOIS).* **35** (4), 1–33 (2017).
6. Wang, H. et al. Shine: Signed heterogeneous information network embedding for sentiment link prediction.Proceedings of the eleventh ACM international conference on web search and data mining. : 592–600. (2018).
7. Cao, Y. et al. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences.The world wide web conference. : 151–161. (2019).
8. Zhang, F. et al. Collaborative knowledge base embedding for recommender systems.Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. : 353–362. (2016).
9. Wang, H. et al. DKN: Deep knowledge-aware network for news recommendation.Proceedings of the 2018 world wide web conference. : 1835–1844. (2018).
10. Wang, H. et al. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems.Proceedings of the 27th ACM international conference on information and knowledge management. : 417–426. (2018).
11. Lin, Y. et al. Learning entity and relation embeddings for knowledge graph completion.Proceedings of the AAAI conference on artificial intelligence. **29**(1) (2015).
12. Wang, Z. et al. Knowledge graph embedding by translating on hyperplanes.Proceedings of the AAAI conference on artificial intelligence. **28**(1). (2014).
13. Ji, G. et al. Knowledge graph embedding via dynamic mapping matrix.Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers). : 687–696. (2015).
14. Yu, X. et al. Personalized entity recommendation: A heterogeneous information network approach.Proceedings of the 7th ACM international conference on Web search and data mining. : 283–292. (2014).
15. Shi, C. et al. Heterogeneous information network embedding for recommendation.. *IEEE Trans. Knowl. Data Eng.* **31** (2), 357–370 (2018).
16. Wang, X. et al. Explainable reasoning over knowledge graphs for recommendation.Proceedings of the AAAI conference on artificial intelligence. **33**(1): 5329–5336. (2019).
17. Sherstinsky, A. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network.. *Phys. D: Nonlinear Phenom.* **404**, 132306 (2020).
18. Kojima, R. et al. KGCN: a graph-based deep learning framework for chemical structures.. *J. Cheminform.* **12**, 1–10 (2020).
19. Wang, H. et al. Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. : 968–977. (2019).
20. Wang, X. et al. Kgat: Knowledge graph attention network for recommendation.Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. : 950–958. (2019).
21. Wang, Z. et al. CKAN: Collaborative knowledge-aware attentive network for recommender systems.Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval. : 219–228. (2020).
22. Li, M., Ma, W. & Chu, Z. KGIE: knowledge graph convolutional network for recommender system with interactive embedding.. *Knowl. Based Syst.* **295**, 111813 (2024).
23. Dwivedi, V. P. et al. Benchmarking graph neural networks.. *J. Mach. Learn. Res.* **24** (43), 1–48 (2023).
24. Hamilton, W., Ying, Z. & Leskovec, J. Inductive representation learning on large graphs.. *Adv. Neural. Inf. Process. Syst.*, **30**. (2017).
25. Kipf, T. N. & Welling, M. Semi-supervised classification with graph convolutional networks.. arXiv preprint arXiv:1609.02907. (2016)
26. Ying, R. et al. Graph convolutional neural networks for web-scale recommender systems. Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining. : 974–983. (2018).
27. Zhang, X. et al. KGCL: A knowledge-enhanced graph contrastive learning framework for session-based recommendation.. *Eng. Appl. Artif. Intell.* **124**, 106512 (2023).
28. Zou, D. et al. Multi-level cross-view contrastive learning for knowledge-aware recommender system.Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval. : 1358–1368. (2022).
29. Yang, Y. et al. Knowledge graph self-supervised rationalization for recommendation.Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining. : 3046–3056. (2023).
30. Wu, J. et al. Self-supervised graph learning for recommendation.Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval. : 726–735. (2021).
31. He, X. et al. Neural collaborative filtering.Proceedings of the 26th international conference on world wide web. : 173–182. (2017).
32. Su, X. & Khoshgoftaar, T. M. A survey of collaborative filtering techniques.. *Adv. Artif. Intell.* **2009** (1), 421425 (2009).
33. Schafer, J. B. et al. Collaborative filtering recommender systems. the adaptive web: methods and strategies of web personalization 291–324 (Springer Berlin Heidelberg, 2007).
34. Vaswani, A. et al. Attention is all you need.. *Adv. Neural. Inf. Process. Syst.*, 30. (2017).
35. Hahnloser, R. H. R. et al. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit.. *Nature* **405** (6789), 947–951 (2000).
36. Memisevic, R. et al. Gated softmax classification.. *Adv. Neural. Inf. Process. Syst.*, 23. (2010).
37. Hamilton, W., Ying, R. & GraphSage Representation learning on large graphs[EB/OL]. (2017).
38. Glorot, X. & Bengio, Y. Understanding the difficulty of training deep feedforward neural networks.Proceedings of the thirteenth international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings, : 249–256. (2010).
39. Kingma, D. P., Ba, J. & Adam A method for stochastic optimization.. (2014). arXiv preprint arXiv:1412.6980.
40. Rendle, S. Factorization machines with libfm.. *ACM Transactions on Intelligent Systems and Technology (TIST)* **3**(3), 1–22 (2012).
41. Rendle, S. et al. BPR: bayesian personalized ranking from implicit feedback.. (2012). arXiv preprint arXiv:1205.2618.
42. Wang, H. et al. Multi-task feature learning for knowledge graph enhanced recommendation.The world wide web conference. : 2000–2010. (2019).

43. Wang, X. et al. Learning intents behind interactions with knowledge graph for recommendation.Proceedings of the web conference 2021. : 878–887. (2021).
44. Meng, Z. et al. Knowledge Graph Cross-View Contrastive Learning for Recommendation.European Conference on Information Retrieval. Cham: Springer Nature Switzerland, : 3–18. (2024).
45. El Alaoui, D. et al. Deep GraphSAGE-based recommendation system: jumping knowledge connections with ordinal aggregation network.. *Neural Comput. Appl.* **34** (14), 11679–11690 (2022).
46. El Alaoui, D. et al. Collaborative filtering: comparative study between matrix factorization and neural network method.Networked Systems: 8th International Conference, NETYS 2020, Marrakech, Morocco, June 3–5, 2020, Proceedings 8. Springer International Publishing, : 361–367. (2021).
47. El Alaoui, D. et al. Overview of the main recommendation approaches for the scientific articles.International Conference on Business Intelligence. Cham: Springer International Publishing, : 107–118. (2021).
48. El Alaoui, D. et al. *Contextual Recommendations: Dynamic Graph Attention Networks with Edge adaptation.* (IEEE Access, 2024).
49. El Alaoui, D. et al. Social recommendation system based on heterogeneous graph attention networks.. *Int. J. Data Sci. Analytics*, : 1–17. (2024).
50. El Alaoui, D. et al. Comparative Study of Filtering Methods for Scientific Research Article Recommendations.. *Big Data Cogn. Comput.*, **8**(12): 190. (2024).
51. El Alaoui, D. et al. A novel session-based recommendation system using capsule graph neural network.. *Neural Netw.*, : 107176. (2025).
52. Yang, Y. et al. Integrating fuzzy clustering and graph Convolution network to accurately identify clusters from attributed graph. *IEEE Trans. Netw. Sci. Engineering* (2024).

Author contributions

Zhong Hua: Conceptualization, Methodology, Writing-original draft. Jianbai Yang: Validation. WeiDongJi: Supervision.

Funding

This work was supported in part by the Natural Science Foundation of Heilongjiang Province of China under Grant (PL2024F007).

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to J.Y.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.