scientific reports

Check for updates

OPEN

Enhancing knowledge graph recommendations through deep reinforcement learning

Jinlian Zhou[1,2]✉, Derong Shen[1], Ying Guo[2], Yanli Zhao[2] & Huijuan Zhang[3]

Recommendation systems are an important tool for information filtering, which have been widely applied in both industry and academia. Although recommendation methods that combine deep learning with collaborative filtering have improved recommendation performance to some extent, issues such as the cold start problem and lack of interpretability remain major challenges. To address these issues, this paper proposes a novel algorithm, RKGnet, a knowledge graph-based recommendation framework using deep reinforcement learning. RKGnet leverages the structural advantages of knowledge graphs and the adaptive decision-making capability of reinforcement learning. By dynamically iterating user preferences within the knowledge graph, RKGnet uncovers the hierarchical latent interests of users and dynamically selects relevant entities through reinforcement learning. It then adapts the recommendation strategy and improves recommendation effectiveness through iterative optimization, enhancing recommendation accuracy and system interpretability. Experimental results demonstrate higher performance when compared with existing methods. RKGnet demonstrates significant advantages in terms of accuracy, robustness, and interpretability, highlighting its broad application prospects in recommendation systems.

Keywords Knowledge graph, Reinforcement learning, Proximal policy optimization, Recommendation system

The rising amount of information and services over the internet has given people a great variety of options as they are able to access music, news, restaurants, movies and books online. Recommendation systems are designed to address this issue of information overload by showing a few items that suit the personalized preferences of the users[1]. Nonetheless, traditional Graph Neural Networks (GNNs)-based recommendation systems are not well-suited to handle the increasing user requirements. Specifically, cold-start problems and the uninterpretability of the recommendation systems are also still important issues. In order to adequately deal with these problems, recent studies have been inclined towards the incorporation of novel model architectures that combine the knowledge information and user-item interaction data to recommend systems. An example is the integration of deep learning methods with knowledge graphs (KGs) into single frameworks to address cold-start challenges and increase the understanding of the system. Some researchers are investigating the applications of new model architectures to integrate the information on knowledge with the user-item interaction information in different systems of recommendations. The most widespread ones are convolutional neural networks (CNN), reinforcement learning (RL), and KGs[2].

We propose RKGnet, a KG-based recommendation framework that leverages RL for personalized recommendations to address the above limitations,. The framework takes user-item pairs as input and predicts the probability of a user interacting with a given item. The core idea behind RKGnet is preference propagation. Specifically, for each user, RKGnet uses their historical preferences as a starting point (seed set) within the KG. It then iteratively expands the user's interests along the links of the KG, dynamically discovering their layered potential interests relative to the candidate items. This process of preference propagation allows the system to adapt and refine recommendations in real time, ensuring that user preferences are accurately captured and updated. By integrating KG and RL, the proposed method improves recommendation performance and offers enhanced interpretability. KG provides structured semantic information that helps the system understand the context of user-item interactions, while RL enables the system to continuously adjust recommendation strategies based on user feedback. This approach mitigates the cold-start problem by leveraging the rich knowledge available in the KG, even when user behavior data is sparse. The cold start dilemma is a serious issue

[1]School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China. [2]School of Computer Science and Engineering, Ningxia Institute of Science and Technology, Shizuishan 753000, China. [3]Yinchuan Maternal and Child Health Hospital, Ningxia 750000, China. ✉email: 543518214@qq.com

in recommendation engines, especially when products or customers do not have enough interaction history. To address this problem, we offer a framework that alleviates it by using item titles and textual features, and utilizing KG to enrich recommendations. The addition of textual data would improve the capacity of the model to make such recommendations even when no previous interaction data is available.

RKGnet is a knowledge graph exploration approach based on path, which enables more dynamic and detailed user preference modeling than traditional embedding-based approaches. Second, RKGnet also features a hybrid reward system, a combination of sequence-level and knowledge-based rewards, which increases its performance with both short and long-term interactions of users. Lastly, knowledge graphs are also used to maximize the accuracy of recommendations, and increase interpretability, providing a clear understanding of how the recommendations are made.

Our contributions in this paper are as follows:

1. The RKGnet algorithm is proposed, a deep reinforcement learning (DRL) algorithm using a KG to solve the cold-start problem and enhance interpretability in recommendation system. RKGnet automatically discovers hierarchical potential interests of users by iterating user preferences in KG.
2. We propose a novel k-step reward function that strategically combines sequence-level and knowledge-level rewards to better balance immediate user engagement with long-term informational value.
3. Extensive experiments were conducted to evaluate the effectiveness of the proposed RKGnet. Based on experimental results, the RKGnet outperformed the existing methods.

Related work

Recommendation systems enhanced by knowledge graphs (KGs) have gained significant attention for their ability to provide richer contextual information, thus improving recommendation accuracy. Existing literature can be categorized into three primary approaches: KG-based recommendation methods, Reinforcement Learning (RL)-based recommendation methods, and Hybrid KG-RL approaches.

Knowledge graph-based recommendation methods

Embedding-based methods

The embedding-based methods obtain the vector representation of entities and relationships in the KG through graph embedding, usually using the graph embedding method of the Trans series. Some researchers have designed a memory network-based structure to solve the problem that RNN networks cannot remember a long time series[3]. When the RNN network iterates the information of each time node, it can better remember and update the entity semantic representation information obtained by using TransE, thus effectively improving the utilization effect of the semantic information of the KG[4]. Thus, the recommendation results are optimized. Some researchers describe music recommendation as a knowledge base completion task to provide individual users with personalized music recommendation[5]. This approach treats users as special entities in the knowledge base related to artists/albums, and uses the knowledge embedding algorithm TransE to get vector representations of users and objects. Tang used the associated entity representations in the historical user interaction data obtained by TransR, and learned their weighted sum by using the self-attention mechanism, to obtain user characteristics more effectively[6]. He et al. further expanded the heterogeneous information network diagram by adding some attributes of users as nodes, such as gender, age, and occupation, to build the edge relationship between users and attributes[7]. Then, TransD method was used for graph embedding learning, thus solving the cold start problem of both users and items[8]. These methods have been evaluated across various frameworks, including those using graph embeddings like TransE and path-based strategies such as RippleNet[9]. These methods focus on embedding entities and relationships within a knowledge graph, improving recommendation accuracy by considering the semantic relationships between users and items.

Path-based methods

The path-based approach to applying knowledge graphs (KG) in recommendation systems primarily focuses on mining various connections between users and items through graph structures[10]. The paths between nodes can be analyzed to extract additional information in the KG[11]. This approach primarily focuses on associative paths among users as well as users and items and uses these paths to derive the corresponding degree between users and items in the KG[12,13]. The major part of this strategy is the ability to capture the user preference paths and the ability to design efficient path propagation and path mining processes in order to capture the user preferences. Path-based schemes provide a superior interpretability in comparison with vector embedding schemes. An example is that other researchers investigate user-friendly neighborhood information to provide individual suggestions. An example is RippleNet where a user clicks an item and the item becomes the "seed node" and builds a ring of first ripple is formed by using the seed node as the head node in triples, and the entities that are connected to it are the ripple entities of the first ripples[9]. Other studies also incorporate KG information using various path-based strategies[14,15]. It is in contrast to the traditional recommendation tasks, which make use of the time of historical user interaction information during path construction at each time node, and in this way, prevents information leakage. The user interaction data sequence is modeled by using self-attention network and predicts the activity[16]. The rule induction and rule guidance-based neural network recommendation system was suggested by M et al. and can be generalized according to the item-related KG, deduces additional relationships between various items, and can generate a recommendation system, which is simple to understand and interpret[17]. Other researchers suggested a policy-guided path reasoning (PGPR) approach, which converts the recommendation problem into a path finding problem with the user as the starting node and the item as the terminal node in the KG, and then offers the recommendation reasons based on the outcome of the path finding problem[18].

Reinforcement learning-based recommendation methods

Reinforcement learning (RL) consists of an adaptive approach using an exploration-interaction generation approach, and its powerful exploration can be applicable to the search of a recommendation system. Research by Google about the use of RL in the recommendations of YouTube brings hope to the industry. The approach of Chen has received the largest revenue of a single online operation of Youtube in last two years, and one of the main accents is the offer of a Top-K Off-Policy amendment scheme[19,20]. Action space of Policy-Gradient algorithm can be used in many online recommendation systems on YouTube. With regards to search result ranking, Ali introduced a ranking technique which maximizes the expected cumulative reward by implementing RL based on taking into consideration the association of non-synchronous ranking results. Based on this, the researchers are able to establish a strong connection between the sequencing of various actions, more than being independent of one another, because the author shows the need to maximize the cumulative payoff[21]. A recommendation framework grounded on DRL has been suggested by the University of Pennsylvania and Microsoft Research Asia to provide customized news suggestions. This recommendation model not only takes into consideration the feedback of the users on the existing recommendations, but also, consider the long term effects of the recommendations on the users[22]. Given that the news is time-sensitive, and the interest of users in news will dynamically evolve with time, authors apply DQN to simulate the temporal shift of the news suggestions, and the future expectation rewards of the users. Simultaneously, user clicks/no clicks and user activity can serve as the indicators of how satisfied users were with the result of the recommendation in order to enhance recommendations accuracy[23,24]. The author Jing Dong suggested a page-wise recommendation system using DRL to recommend the next page based on the instant feedback of users and to put items on the correct position in a page that decides the position of each item when creating a page of recommended items[25]. Serial interaction behavior between users and the recommendation system was modeled by Zhao using a Markov decision process. A recommendation system with automatic feature extraction and strategy trial-and-error optimization using DRL technology was developed, and the model's parameters were pre-trained using simulators before implementation[26]. The reliability of the suggestion is enhanced.

Hybrid KG-RL methods

However, while these approaches improve accuracy, they still face challenges such as the cold start problem and lack of interpretability. Recent literature has tried to deal with such problems using RL. Google came up with RL-based dynamic ranking on YouTube recommendations as an example[19]; however, these models normally do not provide the interpretability of why some recommendations are made. Conversely, our method, RKGnet, combines deep reinforcement learning (DRL) with a knowledge graph to solve both the cold start problem and interpretability. RKGnet takes advantage of the structural benefits of knowledge graphs to represent user-item interaction and apply DRL to explore and optimize user preferences. This allows it to select the entities in the knowledge graph that are relevant dynamically, providing better recommendations accuracy and transparency in decisions-making. RKGnet is better than the previous models, e.g., TransE[4], RippleNet[9], etc., as it tackles the problem of accuracy and interpretability. Compared to embedding-based models, RKGnet models user-item connections as search paths inside the knowledge graph, which uncovers multi-level latent user interests and provides users with interpretable feedback into the recommendation process. Experimental results confirm these improvements since RKGnet was subjected to significant improvements compared to baseline models based on both recommendation accuracy and interpretability of the system.

However, our method, RKGnet, combines DRL and KG to enhance the accuracy and interpretability of recommendations. RKGnet represents the interaction between items and users as search paths within the KG, and RL actively chooses the useful entities and optimizes the recommendation strategy with time. This combined solution enables RKGnet to solve cold-start problems and improve system understanding more than current solutions such as TransE or RippleNet.

Problem formulation

The notations used in this paper and their corresponding descriptions are summarized in Table 1.

The recommendation can be formulated as follows. Suppose $U = \{u_1, u_2, \ldots\}$ represents the set of users, and $V = \{v_1, v_2, \ldots\}$ represents the set of items, in a classical recommendation setting. According to users' implicit feedback, the user-item interaction matrix $Y = \{y_{uv} \mid u \in U, v \in V\}$ can be defined as follows:

$$y_{uv} = \begin{cases} 1, & \text{if the user uinteracted with the item}v \\ 0, & \text{otherwise} \end{cases}$$

. The binary indicator $y_{uv} = 1$ denotes an implicit user-item interaction (e.g., clicks, views, or browsing). Alongside the interaction matrix Y, we leverage a (KG) composed of $(\underline{h}, \underline{r}, \underline{t})$ triples, where $\underline{h}$ and $\underline{t} \in E$ are head/tail entities, $\underline{r} \in R$ is their relation, and E and R represent the entity and relation sets in the KG, respectively.

We map the recommendation system to the KG, and model the recommendation search path as a Markov problem to facilitate the introduction of reinforcement learning (RL). A RL network based on strategy is mainly composed of two key parts: the state network and the strategy network. An agent is a virtual entity that searches for events defined according to task requirements, and its purpose is to find the optimal path and to achieve the maximum reward. In this paper, RL network is applied to the KG, and the KG is modeled as a Markov network. To better characterize the nodes in the graph, the definition of agent in KGPolicy[27] is adopted, and the node closest to the target needs to be identified through the agent to achieve accurate characterization. In short, the KG-based RL network policy recommendation method is a path-oriented and efficient recommendation learning method that relies on agent state transfer and then learns through a policy network. The detailed

Notation	Description	Notation	Description
Y	User-item interaction	G	Knowledge graph
U	User set	V	Item set
E	Entity set	R	Relation set
S	State set	A	Action set
P	Transition probability	u	User
v	Item	$\underline{h}$	Head entity
$\underline{r}$	Relation	$\underline{t}$	Tail entity
s	State	a	Action
r	Reward	t	Time
ϵ	Margin parameter	e	Embedding
γ	Discount factor		
$P(s_{t+1}\|s_t, a_t)$	Transition	$R(s_t, a_t)$	Reward function
$O(s)$	Observation function	$V_w^{\pi}(o_k)$	Value function
$\pi_\theta(\boldsymbol{u}_k \mid \boldsymbol{o}_k)$	Policy	$\nabla J(\Theta)$	Policy gradient

Table 1. Notations and their description.

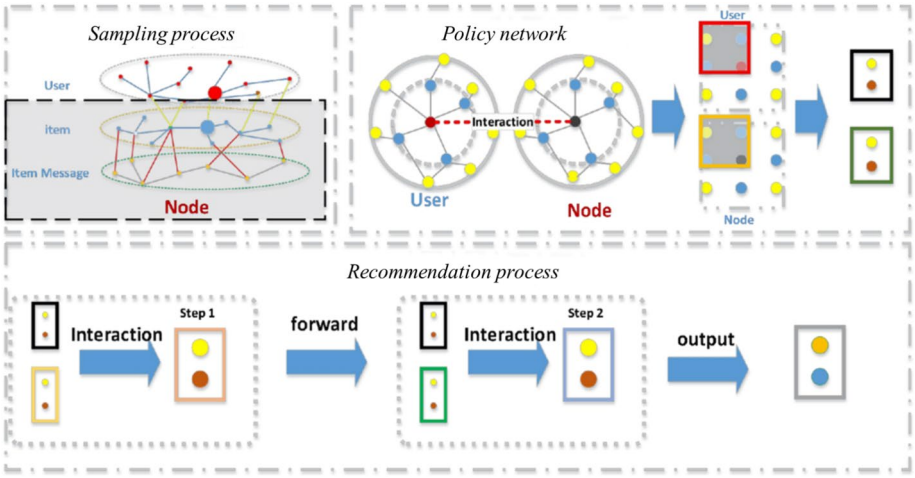

Fig. 1. The processing of RKGnet.

workflow of, RKGnet, our framework is illustrated in Fig. 1. Sampling process abstracts the data set sampling into nodes of the KG, then the policy network carries out Markov modeling on the KG to generate a model training environment suitable for RL, and finally a feedback technique is adopted to tune the parameters of the RL model to realize the recommendation system.

Proposed method

We utilize items' titles and other textual metadata as key features within our recommendation model to deals with the cold start problem. By embedding these textual features into a KG, we can generate meaningful recommendations for new items. This allows our model to make more accurate predictions even when interaction data is sparse. The recommendation system is mapped to a KG where both item and user data is embedded in the graph nodes. Specifically the items are represented as nodes and their association (e.g. similarity, co-occurrence) are viewed as an edge in the graph. The user interaction and other metadata (e.g., item type) are then mapped onto the graph making it rich and full of context that supports the generation of recommendations. After mapping the recommendation system to the KG, search path modeling between the items is created. The KG represents a semantic space upon which items relate to each other in a number of different relationships like category, attributes, or past interactions with users. Such relationships enable the system to identify not only items that are directly related but also those that are indirectly related, as well as present a wider and contextually more relevant list of recommendations.

Knowledge graph

KG represents human knowledge in the form of a directed graph, with entities representing nodes and their relationship being represented by the type of edge. The directed edge is an expression of a relation between two

entities, one the head and the other a tail, which is usually denoted as a triplet $(\underline{h}, \underline{r}, \underline{t})$. These triplets show that a specific relation $\underline{r}$ exists between the head $\underline{h}$ and the tail $\underline{t}$. Although the use of symbolic triplets has a very clear structure, it may be challenging programmatically[11]. To overcome it, we use the methods of knowledge graph embedding (KGE) that transforms discrete relations and entities to continuous vectors. Our method is based on TransE[13], a popular KGE algorithm that represents the relationships as linear translations in the embedded space. TransE represents the connection between tail entity $\underline{t}$ and head entity $\underline{h}$ through relation $\underline{r}$ using the following translation principle.

$$\mathrm{h} + \mathrm{r} \approx \mathrm{t},$$

In the equation above, the $\mathrm{h}, \mathrm{r}, \mathrm{t} \in \mathbb{R}^{dim}$ are the dim-dimensional embeddings of the respective components, as shown in Fig. 2. The scoring function used to evaluate the likelihood of a triplet's validity is defined as:

$$g(\underline{h}, \underline{r}, \underline{t}) = -\|\mathrm{h} + \mathrm{r} - \mathrm{t}\|_{1/2},$$

The greater the score, the greater the likelihood of the triplet $\underline{h}, \underline{r}, \underline{t}$ to be valid. The distance is usually determined by the L1 or L2 norm. According to this scoring task, TransE uses a margin-based loss to learn the embeddings, which is defined as:

$$L = \sum_{(\underline{h}', \underline{r}, \underline{t}') \in S'} \mathrm{ReLU}(\mu + g(\underline{h}', \underline{r}, \underline{t}') - g(\underline{h}, \underline{r}, \underline{t})),$$

where $\mu > 0$ denotes the margin parameter. The triplet $(\underline{h}, \underline{r}, \underline{t})$ represents a valid training example, whereas $(\underline{h}', \underline{r}, \underline{t}')$ refers to a corrupted or negative sample. The goal of the loss function is to increase the difference between the scores assigned to correct triplets and those assigned to negative ones. These negative samples are typically generated by randomly substituting either the tail or the head entity in a valid triplet. During training, TransE employs gradient descent to iteratively update the embedding vectors in a way that minimizes the loss and enhances the quality of the learned representations.

The framework of RKGnet

In the framework of RL, an agent acquires knowledge through episodic interaction with an environment, aiming to master a task by acquiring a policy that translates observations into actions. Based on the encoder described above, the framework of RKGnet is shown in Fig. 3.

In scenarios involving continuous state and action spaces, a Markov Decision Process (MDP) is typically defined by a state space S, an action space A, a state transition function $P(\boldsymbol{s}_{t+1} \mid \boldsymbol{s}_t, \boldsymbol{a}_t)$, and a reward function $r = R(\boldsymbol{s}_t, \boldsymbol{a}_t)$, where $\boldsymbol{s} \in S$, $\boldsymbol{a} \in A$, and r denotes a scalar reward signal. In more complex settings where the agent cannot fully observe the true environment state, the model extends to a partially observable MDP (POMDP). In this case, the true state $\boldsymbol{s}$ is hidden, and the agent receives an observation o generated by an observation function $O(\boldsymbol{s})$, which maps the hidden states to observable outputs. This POMDP framework is particularly useful in situations where data come from sensors, as is the case in this study. For the sake of consistency, we refer to both partially and fully observable environments under the umbrella of POMDPs throughout this paper, treating an MDP as a special instance where the observation function simply returns the state itself (i.e., an identity mapping).

To build our task, we adopt an MDP framework, where an agent sequentially interacts with an environment through discrete time steps. At each step t, the process resides in a specific state $s_t \in S$,where the state represents all relevant information necessary to make sequential recommendations. This includes the user's click

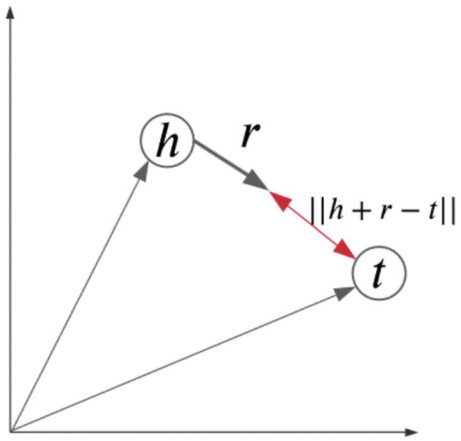

Fig. 2. TransE in 2-D space.

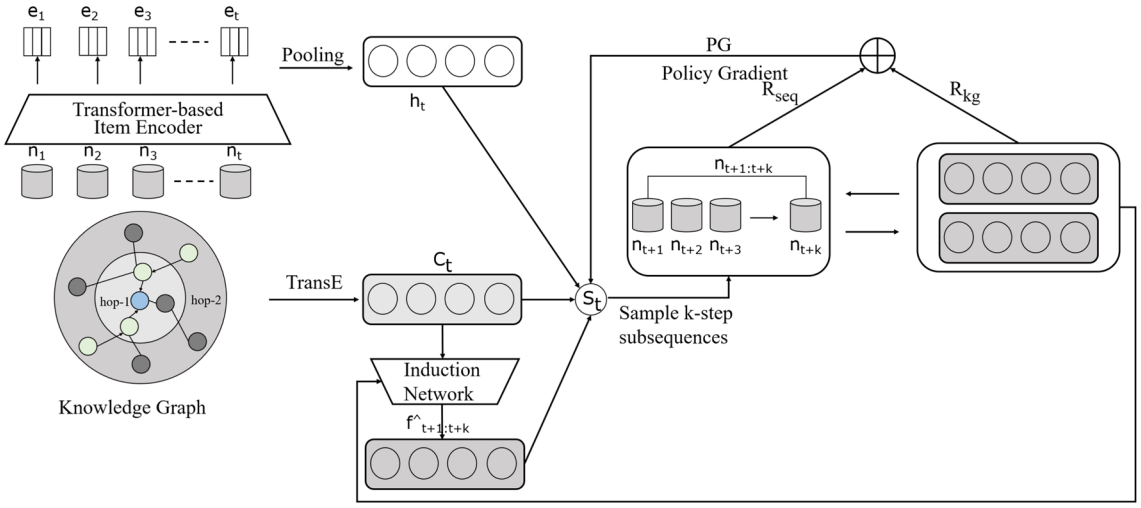

Fig. 3. The framework of RKGnet.

interaction history as well as the KG information associated with the items they have interacted with. Therefore, the state at time t can be defined as:

$$s_t = [e_{1:t}, G]$$

where $e_{1:t}$ indicates the historical items clicked by the u, and G indicates the KG information contained the historical items. The initial status is set to $s_0 = [\varnothing, G]$, where $\varnothing$ indicates that the interactive item is empty. We used an embedded vector $\mathrm{e}_{s_t} \in \mathbb{R}^{dim}$ to encode the state s_t information. e_{s_t} is used to encode important information showing the state s_t.

In current state s_t, agent takes an action $a_t \in A$ by selecting a recommendation item n_{t+1} from the set of candidates N. Action behavior can be modeled by the policy function $\pi(s_t)$, which is a mapping from the state s_t to the probability distribution of all possible items. The softmax function is adopt to calculate the probability of a possible interaction with an item.

$$\pi(a_t \mid s_t) = \frac{\exp\left\{e_{n_j} W_1 \mathrm{e}_{s_t}\right\}}{\sum_{n_j \in N} \exp\left\{e_{n_j} W_1 \mathrm{e}_{s_t}\right\}}$$

where e_{n_j} represents the representation of the j item, W_1 is the parameter in the bilinear product, and e_{s_t} is the representation of the state s_t. After each action, agent receives an intermediate reward value of r_{t+1}. We set the reward function $R(s_t, a_t)$ to reflect the recommended performance required in our task. In addition, we use the state transition function $T(T : S \times T \to S)$ to update the state;

$$s_{t+1} = T(s_t, a_t) = T([u, e_{1:t}, G], \mathrm{e}_{j(a_t)})$$

the new state s_{t+1} can be written as $[u, e_{1:t+1}, G]$, and it is also associated with an embedded vector $\mathrm{e}_{s_{t+1}}$. The encoder mentioned here corresponds to the "Encoder" part shown in Fig. 3. The item encoder utilizes a Transformer model to encode the item sequence, and its architecture is detailed in Fig. 4.

The representation vector generated by the item encoder is denoted as e_i. The encoder is composed of three layers. The first layer is a word layer, which converts the item's description (such as title, attribution) into low-dimensional vectors. The second is an MHA-based layer, which is responsible for capturing and learning the attention between the words. The third one is an additional attention network, which is used to model the relative importance of different words. The vectors of the previous layer are calculated by attention and aggregated to generate the representation of the item's title e_t. Because the category information of the item selected by the user may also show their preference, the item's category is modeled by a learnable transformation matrix, and the representation of the matrix is expressed as e_c. Finally, item is represented as an vector $e = [e_t; e_c]$ formed by concatenating its title representation and category representation.

Proximal policy optimization

Since PPO has shown superior performance in many RL tasks, we used PPO as a policy gradient method. It is designed to approximate the trust region policy optimization approach by incorporating a constraint on policy adjustments through a clipped objective function. In PPO, the objective function is formulated using the probability ratio $p_k(\theta)$, which is formulated by;

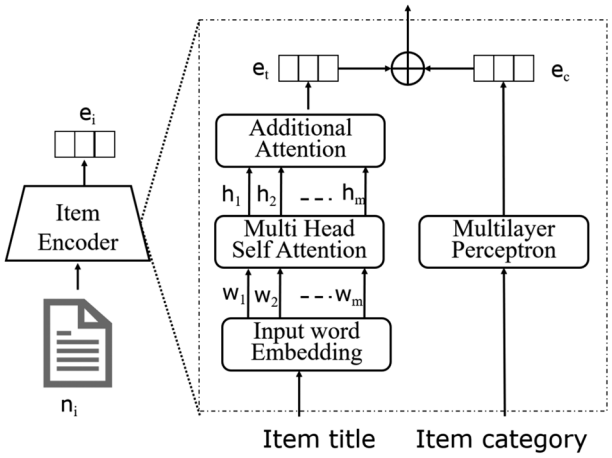

Fig. 4. Encoder model.

$$p_k(\theta) = \frac{\pi_\theta(\boldsymbol{a}_k \mid \boldsymbol{o}_k)}{\pi_{\theta\ \mathrm{old}}(\boldsymbol{a}_k \mid \boldsymbol{o}_k)}$$

.

The fundamental concept involves creating two surrogate objectives. Firstly, the probability ratio $p_k(\theta)$ and the benefits $A_w^\pi(\boldsymbol{o}_k, \boldsymbol{a}_k)$, and the another objective is a clipped version (using a clipping parameter ϵ) of $p_k(\theta)$ multiplied by $A_w^\pi(\boldsymbol{o}_k, \boldsymbol{a}_k)$. The optimization objective $J(\theta)$ is defined as the expected value, over policy-induced trajectories, of the minimum between these objectives.

$$\mathrm{obj}_1 = p_k(\theta)A_w^\pi(\boldsymbol{o}_k, \boldsymbol{a}_k); \qquad \mathrm{obj}_2 = \mathrm{clip}(p_k(\theta)A_w^\pi(\boldsymbol{o}_k, \boldsymbol{a}_k), 1-\varepsilon, 1+\varepsilon)$$
$$J(\theta) = \mathbb{E}_{p(\tau)}[\min(\mathrm{obj}_1, \mathrm{obj}_2)]$$

.

The clipped objective function inherently maintains an upper bound on the Kullback–Leibler (KL) divergence between consecutive policy distributions during optimization. This property contributes to convergence by preventing drastic changes in the policy between updates. In our PPO implementation, we estimate advantages by computing the difference between a learned value function and empirical return. Here, $r(\cdot)$ and γ represent the reward function, and discount factor, respectively, defined as:

$$A_w^\pi(\boldsymbol{o}_k, \boldsymbol{a}_k) = [\sum_{\ell=k}^{T} \gamma^{\ell-k} r(\boldsymbol{o}_\ell, \boldsymbol{a}_\ell)] - V_w^\pi(\boldsymbol{o}_k)$$

here, the learned value function V_w^π is achieved using the cost function as follow:

$$L(\boldsymbol{w}) = \sum_{i=1}^{M} (V_w^\pi(\boldsymbol{o}_k^i) - [\sum_{\ell=k}^{T} \gamma^{\ell-k} r(\boldsymbol{o}_\ell^i, \boldsymbol{a}_\ell^i)])^2$$

. In practice, policy gradient algorithms (PGA) are used to update the policy by utilizing a batch of trajectories (rollouts) obtained through agent-environment interaction. Each trajectory corresponds to a single episode, and a sample from a trajectory collected at step k includes action $\boldsymbol{a}_k$, observation $\boldsymbol{o}_k$, and reward $r_k(\boldsymbol{o}_k, \boldsymbol{a}_k)$. The update process involves performing a gradient ascent in θ and a gradient descent in $\boldsymbol{w}$. The update equations can be expressed as:

$$\boldsymbol{w}^+ = \boldsymbol{w}^- - \beta_{\boldsymbol{w}} \nabla_{\boldsymbol{w}} L(\boldsymbol{w})|_{w=w^-}; \qquad \theta^+ = \theta^- + \beta_\theta \nabla_\theta J(\theta)|_{\theta=\theta^-}$$

the β_w and β_θ denote learning rates for the value function $V_w^\pi(o_k)$ and policy $\pi_\theta(\boldsymbol{a}_k \mid \boldsymbol{o}_k)$, respectively.

Reward function based on KG

For RL algorithms, defining a suitable reward function is particularly important. In item recommendations, final performance is often calculated based on the exact match of item IDs. At the same time, interactive sequences are generated by users clicking on the item content of interest. Therefore, in addition to item-level performance, it is also important to measure how good or bad the inferred knowledge-level preferences are. Based on the above description, in the time step t, we define the k-steps reward function by integrating two different reward functions:

$$R(s_t, a_t) = \lambda R_{kg}(n_{t+1:t+k}, \hat{n}_{t+1:t+k}) + (1-\lambda) R_{seq}(n_{t+1:t+k}, \hat{n}_{t+1:t+k})$$

where, $R_{seq}(\cdot,\cdot)$ and $R_{kg}(\cdot,\cdot)$ represent the sequence-level and knowledge-level of reward to the user click true reading subsequence $n_{t+1:t+k}$ and recommend the subsequence of $\hat{n}_{t+1:t+k}$ as the information input , and λ is a hyperparamter to balance $R_{seq}(\cdot,\cdot)$ and $R_{kg}(\cdot,\cdot)$. It should be noted that only the input of the k -steps is considered here to approximate the overall performance. Then we discuss how to calculate $R_{seq}(\cdot,\cdot)$ and $R_{kg}(\cdot,\cdot)$.

An effective reward function evaluates the performance at each step as well as the overall quality across the entire recommendation sequence in sequence recommendation. Drawing inspiration from machine translation evaluation, we adopt the BLEU metric[28] to assess sequence recommendations. Formally, the interaction of a given actual subsequence $n_{t+1:t+k}$ and recommend subsequence of $\hat{n}_{t+1:t+k}$, will reward function is defined as:

$$R_{seq}(n_{t+1:t+k}, \hat{n}_{t+1:t+k}) = \exp(\frac{1}{M}\sum_{m=1}^{M} \log \mathrm{prec}_m)$$

where $\mathrm{prec}_m$is the corrected precision value, computed as follows:

$$\mathrm{prec}_m = \frac{\sum_{p_m \in n_{t+1:t+k}} \min(\#(p_m, n_{t+1:t+k}), \#(p_m, \hat{n}_{t+1:t+k}))}{\sum_{p_m \in n_{t+1:t+k}} \#(p_m, n_{t+1:t+k})}$$

where ρ_m is real click record $n_{t+1:t+k}$ m-gram subsequence, $\#(p_m, n_{t+1:t+k})$ is the quantity of occurrences of ρ_m in $n_{t+1:t+k}$. M decides how many M-gram precision values to use. Using the above formula, it is possible to observe that this kind of reward function can help the recommendation algorithm create a more stable m-gram based on the actual sequence. Therefore, it is natural to measure sequence-level performance in tasks.

In the second reward function, the exact match to the item number is not concerned. Instead of evaluating recommendations directly at the item level, we focus on assessing the quality of the knowledge-level features represented in the sequence. Given a ground-truth subsequence and a predicted subsequence, denoted by $n_{t+1:t+k}$ and $\hat{n}_{t+1:t+k}$, respectively, we adopt a straightforward averaging approach to aggregate the TransE embedding vectors. This allows us to obtain knowledge-level representations for the subsequences, represented as $c_{t+1:t+k}$ and $\hat{c}_{t+1:t+k}$. These two representations of knowledge levels indicate the preference or liking of the user for the knowledge and logic concerning the item. Cosine similarity is taken as the reward in order to measure the difference between the two vectors:

$$R_{kg}(n_{t+1:t+k}, \hat{n}_{t+1:t+k}) = \frac{c_{t+1:t+k} \cdot \hat{c}^T_{t+1:t+k}}{\|c_{t+1:t+k}\| \cdot \|\hat{c}_{t+1:t+k}\|}$$

.

Learning algorithm

This work proposes a RL approach augmented with external KG information for sequential recommendation. The main challenge lies in constructing meaningful knowledge-aware state representations that effectively capture evolving user preferences. To address this, we employ Monte Carlo sampling to generate truncated interaction subsequences for MDP training, coupled with a novel pairwise ranking mechanism that enables inductive learning in the recommendation network. The update pattern of the RL network is shown in Figure 5.

In combination with the task of news recommendations, our goal is to learn a random strategy π that maximizes the expected cumulative reward $J(\Theta)$ for all users. The derivative of $J(\Theta)$ is:

$$\nabla J(\Theta) = E_\pi[\sum_u \sum_{j=t}^{n} \gamma^{j-t} R_j \frac{\nabla \pi(a_t \mid s_t; \Theta)}{\pi(a_t \mid s_t; \Theta)}]$$

where the discount factor is represented by γ which balances the importance of current and future rewards, and Θ represents all the parameters that need to be learned[29]. A truncated strategy gradient training is used to learn the parameters. In particular, for each state s_t , model simulates a truncated k-steps sequence $\hat{n}^{(l)}_{t+1:t+k}$, given $\hat{n}^{(l)}_{t+1:t+k}$, the learning process is:

$$\nabla\Theta = \sum_{j=t}^{t+k} \gamma^{j-t} R_j \frac{\nabla \pi(\hat{n}^{(l)}_t \mid s_t; \Theta)}{\pi(\hat{n}^{(l)}_t \mid s_t; \Theta)}$$
$$\Theta \leftarrow \Theta + \alpha \nabla \Theta$$

.

The model iteratively performs this process L times to refine its estimate. In the network training phase, a crucial component is the inductive network, which forecasts future preferences from current ones. To train this neural network, a straightforward approach is to optimize it against the ground truth preference vector. However, for recommendation tasks, the KG information may have irrelevant information or noise. Furthermore, in the

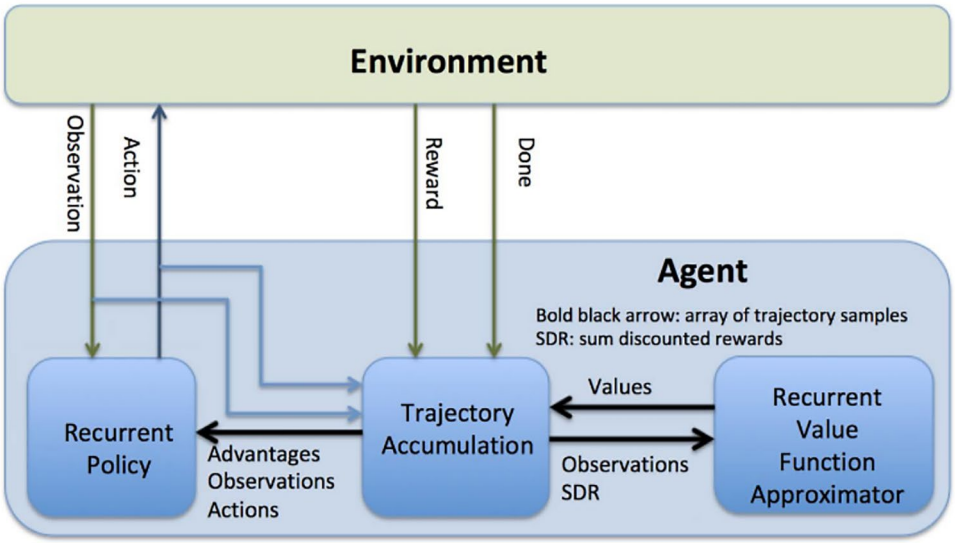

Fig. 5. RL learning algorithm.

sequence recommendation, the regression monitoring information for the real vector is sparse because only one fact-based preference vector $\rho_{t+1:t+k}$ is available for a given subsequence. We introduce a strategy to enhance the training process to better train the inductive network.

Based on state s_t simulation subsequence, $[\rho^1_{t+1:t+k}, \rho^2_{t+1:t+k}, \ldots, \rho^l_{t+1:t+k}]$, we can export their corresponding knowledge representation. By calculating their reward values, paired comparisons can be formed as additional constraints to improve learning. First of all, use of their bonus to construct L is the sequence of preference order, the given$\rho^l_{t+1:t+k}$ and $\rho^{l'}_{t+1:t+k}$, and then add constraint in pairs to train network, when $1 \leq l, l' \leq L$, if $R_{kg}(n_{t+1:t+k}, n^l_{t+1:t+k}) > R_{kg}(n_{t+1:t+k}, \hat{n}^{l'}_{t+1:t+k})$, then $MLP(\hat{\rho}^l_{t+1:t+k}) > MLP(\hat{\rho}^{l'}_{t+1:t+k})$. The core innovation of the model is the inclusion of future knowledge-based preferences $\rho_{t+1:t+k}$. Such a factor was missing from the previous knowledge perception recommendation model, which made it difficult to capture the evolution of user interests. Using the trained model, we can compute the probability that a user will click on a specific candidate item based on their interaction history. These candidate items are then ranked according to their predicted click probabilities, and the top-N items are selected to form the final recommendation list.

Pseudocode and evaluation protocol

To enhance transparency and reproducibility, we provide the following pseudocode that summarizes the training and inference process of RKGnet.

Input: Training dataset (D_{train}), Knowledge Graph (KG), Hyperparameters ($\alpha, \beta, \gamma, \lambda$), Number of epochs ($num\_epochs$), Number of iterations per epoch (L)
Output: Recommendation list for each user (R_{final})

Initialize knowledge graph embeddings using TransE
Initialize policy network (π) and value network (V)
Initialize environment based on user-item interactions and KG
for epoch = 1 to num_epochs **do**
  for batch in D_{train} **do**
    Initialize state s_0 based on user interaction and KG
    for t = 1 to L **do**
      Select action a_t based on policy network $\pi(s_t)$
      Execute action a_t, observe next state s_{t+1} and reward r_{t+1}
      Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$
      Update policy network using Proximal Policy Optimization (PPO)
    end for
    Evaluate performance on validation set
  end for
end for
Output final recommendation list R_{final} based on top-K recommended items for each user

Algorithm 1. Training and inference process for RKGnet.

Computational complexity

The suggested framework is comprised of three main steps, which include preference propagation over the knowledge graph, state representation on the inductive network, and decision-making on the RL. Multi-hop neighbor expansion is part of preference propagation process that has worst-case time complexity of $O(k^h)$ where k is the average number of neighbors and h is the number of hops. The inductive encoder network has a time per user or item embedding of On $O(n \cdot d^2)$, with n nodes and d the dimension of the embedding. The

policy network used by the RL agent has complexity determined by the size of action space and the size of the state dimensions, usually $O(a \cdot d)$, a is a size of action space. In general, RKGnet is a linear scaling computational cost that depends on the number of interactions and entities in the graph, which makes it viable to medium and large-scale knowledge-informed recommendations. Also, the model can be used with batched and parallel processing, which further enhances efficiency during training and inference.

Experiments

In this section, we discuss the baselines, datasets, and experiment setup, followed by the results of the experiment. We also include a selection of the hyperparameters and an ablation study. We conduct experiments on the performance of our framework on cold start by testing items that the user has limited interaction data. In particular, we pay attention to new objects that do not have enough history of interaction, and based on their metadata (e.g., titles, descriptions), we make recommendations. We prove that the suggested framework is superior to classical approaches in cold start cases. In the case of products that have no user interactions before, the item titles and knowledge graph embeddings are much more likely to enhance the precision of recommendations.

Datasets

In the experimental setup, we use three different datasets for knowledge graph-based recommendation, such as Amazon-book, Yelp2018, and Alibaba-iFashion. The statistic of these datasets can be seen in the Table 2.

- Amazon-book composed of book purchasing records from Amazon.com, where items are individual books. Its KG represents connections among books, authors, and categories.
- Yelp2018 is obtained from the Yelp Challenge 2018 and contains user interactions with local businesses, where items represent distinct business establishments such as restaurants and shops.
- Alibaba-iFashion is a fashion dataset from Alibaba's platform, where items correspond to specific fashion products such as clothing and accessories, and it features rich item relationships in its knowledge graph.

Following the approach established in prior work[30,31], we preprocess all datasets by mapping items to knowledge graph entities via name matching and adopt the same knowledge graph construction methodology. For each dataset, we split the interactions by allocating 80% of each user's history to training, 10% of the training set for validation, and the remaining 20% for testing, ensuring consistent experimental conditions.

Evaluation metrics

The performance of the proposed RKGnet and the baseline models is evaluated using three widely-adopted metrics: Recall@K, Precision@K, and Normalized Discounted Cumulative Gain (NDCG@K) with $K = 20$ as the default cutoff. These metrics are defined as follows:

$$\text{Precision@K} = \frac{\sum_{u \in U} |R(u) \cap T(u)|}{\sum_{u \in U} |R(u)|}; \qquad \text{Recall@K} = \frac{\sum_{u \in U} |R(u) \cap T(u)|}{\sum_{u \in U} |T(u)|}$$

$$\text{NDCG@K} = \frac{\text{DCG@K}}{\text{IDCG@K}}; \qquad \text{DCG@K} = \sum_{i=1}^{K} \frac{r_i}{\log_2(i+1)}; \qquad \text{IDCG@K} = \sum_{i=1}^{\min(|T(u)|, K)} \frac{1}{\log_2(i+1)}$$

where $R(u)$ represents the list of top-K predicted items for user u, and $T(u)$ represents the set of ground-truth items for the user, and $r_i \in \{0, 1\}$ denotes whether the item at position i is relevant to the user.

Parameters setting

KG embeddings are initialized using the TransE algorithm with entity/relation dimensions of 32, applying L2 regularization (weight 0.01) to prevent overfitting. The Transformer encoder consists of 6 layers with a hidden dimension of 512, 4 attention heads. The setting of model training parameters is shown in Table 3.

We employ breadth-first search to traverse the knowledge graph and generate 100,000 item triplets through random walks. These triplets are then used to optimize TransE embeddings via a margin-based loss function, enforcing geometric consistency in the vector space. The Transformer encoder pre-trains on product title sequences (truncated to 50 tokens) through self-supervised learning, minimizing cross-entropy loss for predicting the next token in the sequence. The learning rate (η) dynamically adjusts as follows;

	User-item interaction			KG		
Dataset	#Users	#Items	#Interactions	#Entities	#Relations	#Triplets
Amazon-book	70679	24915	847733	88572	39	2557746
Yelp2018	45919	45538	1185068	90961	42	1853704
Alibaba-iFashion	114737	30040	1781093	59156	51	279155

Table 2. Dataset statistics summary.

Parameter	Value	Parameter	Value
clip_ϵ	0.2	discountfactor_γ	0.9
kl_coeff	0.5	balance_λ	0.4
batch_size	4,096	learningrate_η	0.0003
entropy_coeff	0.02	max_grad_norm	0.5
embedding_dimension	32	iteration_number	3
margin_μ	1.0		

Table 3. Hyperparameter configuration.

$$\eta = 3 \times 10^{-4} \quad \text{(initial value)},$$
$$\eta = \eta \times 0.8 \quad \text{every 5 epochs},$$

with early stopping if NDCG does not improve for 10 consecutive epochs. Gradient clipping is applied separately to the policy and value networks during PPO updates to prevent the probability ratio $\rho_k / \pi_{\theta_{\text{old}}}$ from exploding and ensure numerical stability.

Baselines

The following baselines were included in a comprehensive comparative study with the proposed RKGnet. They were selected to represent major research strands, including sequential and session-based models (GRU4RC[32], SR-GNN[33]), knowledge-aware models that integrate graph-side information (DKN[34], KGAT[31], KGIN[35], KGRec[36]), and reinforcement learning-based KG reasoning methods that perform explicit pathfinding (PGPR[37], MultiHopKG, TPGR[38], RKGR[39], ReMR[40], DRLRS[41], EQGPR[42]). This selection ensures a rigorous evaluation against diverse state-of-the-art paradigms.

- *GRU4RC* Models user session sequences using Gated Recurrent Units (GRUs) to predict the next click.
- *SR-GNN* Represents sessions as graphs and uses Graph Neural Networks (GNNs) to capture complex item transitions and higher-order connectivity.
- *DKN* Integrates knowledge graph entity embeddings and word embeddings via a CNN for click-through rate prediction.
- *KGAT* Employs a graph neural network to explicitly model high-order relationships in user–item–KG collaborative knowledge graphs.
- *KGIN* Models finer-grained user–item relations and leverages relational paths for representation aggregation in a GNN.
- *KGRec* Uses self-supervised rationalization to identify informative KG connections and cross-view contrastive learning to mitigate noise.
- *PGPR* A foundational reinforcement learning method that performs policy-guided path reasoning over KGs to generate interpretable recommendations via multi-hop paths.
- *MultiHopKG* An RL-based KG reasoning approach designed to mitigate false negative supervision and reduce sensitivity to spurious paths.
- *TPGR* Employs policy gradient learning to traverse a hierarchical item tree, addressing the challenge of discrete action spaces.
- *RKGR* Incorporates user reviews to better capture preferences and uses predicted ratings as reward signals in reinforcement learning.
- *ReMR* An RL-based framework that integrates ontology-view and instance-view KGs for multi-level reasoning to capture hierarchical user interests.
- *DRLRS* A deep reinforcement learning recommendation system that replaces self-attention with a soft attention mechanism.
- *EQGPR* Jointly optimizes recommendation accuracy and explanation quality by evaluating reasoning paths based on recency, popularity, and diversity.

Overall performance

In this section, we present the overall performance of the proposed model using three different datasets. The overall performance comparison of different recommendation methods across three datasets: Amazon-Book, Yelp2018, and Alibaba-iFashion is shown in Table 4. The performance is evaluated using Recall@20 and Precision@20 for each method, along with the corresponding standard deviation (shown as ±). Methods are ranked by performance, with RKGnet outperforming other models in all three datasets, achieving the highest Recall and Precision values. In particular, RKGnet shows improvements of +1.92%, +2.15%, and +1.59% in Recall@20 for Amazon-Book, Yelp2018, and Alibaba-iFashion, respectively, as well as consistent improvements in Precision@20. The results highlight the effectiveness of RKGnet compared to other baseline models, such as GRU4RC, SR-GNN, and ReMR, which demonstrate lower performance in most cases. The performance improvements indicate that the proposed method offers a robust solution for recommendation tasks, particularly in handling varying datasets.

The comparative performance in terms of NDCG 20 metric of different recommendation techniques on three datasets, Amazon-Book, Yelp2018 and Alibaba-iFashion is shown in Fig. 6. The techniques can be categorized

Method	Amazon-Book		Yelp2018		Alibaba-iFashion	
	Recall@20	Precision@20	Recall@20	Precision@20	Recall@20	Precision@20
GRU4RC	0.1382 ± 0.0023	0.0453 ± 0.0017	0.1256 ± 0.0028	0.0412 ± 0.0019	0.0983 ± 0.0021	0.0356 ± 0.0015
SR-GNN	0.1425 ± 0.0028	0.0468 ± 0.0021	0.1298 ± 0.0032	0.0427 ± 0.0023	0.1027 ± 0.0026	0.0372 ± 0.0018
DKN	0.1293 ± 0.0021	0.0421 ± 0.0016	0.1187 ± 0.0025	0.0389 ± 0.0018	0.0921 ± 0.0019	0.0332 ± 0.0014
KGAT	0.1489 ± 0.0035	0.0489 ± 0.0028	0.1362 ± 0.0038	0.0448 ± 0.0031	0.1030 ± 0.0031	0.0378 ± 0.0025
KGIN	0.1436 ± 0.0031	0.0472 ± 0.0024	0.1321 ± 0.0034	0.0435 ± 0.0026	*0.1208 ± 0.0028*	0.0413 ± 0.0022
KGRec	0.1512 ± 0.0038	*0.0508 ± 0.0031*	0.1398 ± 0.0041	0.0461 ± 0.0034	0.1188 ± 0.0034	0.0429 ± 0.0028
PGPR	0.1463 ± 0.0034	0.0481 ± 0.0027	0.1345 ± 0.0037	0.0442 ± 0.0030	0.1072 ± 0.0030	0.0387 ± 0.0024
MultiHopKG	0.1441 ± 0.0032	0.0473 ± 0.0025	0.1328 ± 0.0035	0.0437 ± 0.0028	0.1058 ± 0.0029	0.0381 ± 0.0023
TPGR	0.1492 ± 0.0036	0.0490 ± 0.0029	0.1371 ± 0.0039	0.0451 ± 0.0032	0.1097 ± 0.0032	0.0396 ± 0.0026
RKGR	0.1528 ± 0.0041	0.0502 ± 0.0034	0.1406 ± 0.0043	0.0463 ± 0.0037	0.1163 ± 0.0037	0.0419 ± 0.0031
ReMR	*0.1562 ± 0.0043*	0.0505 ± 0.0036	0.1415 ± 0.0045	*0.0475 ± 0.0039*	0.1172 ± 0.0039	0.0423 ± 0.0033
DRLRS	0.1519 ± 0.0039	0.0499 ± 0.0032	0.1392 ± 0.0042	0.0458 ± 0.0035	0.1158 ± 0.0035	0.0417 ± 0.0029
EQGPR	0.1558 ± 0.0045	0.0512 ± 0.0038	0.1445 ± 0.0047	0.0471 ± 0.0041	0.1191 ± 0.0041	*0.0435 ± 0.0035*
RKGnet (Ours)	**0.1592** ± 0.0028	**0.0523** ± 0.0021	**0.1468** ± 0.0031	**0.0483** ± 0.0023	**0.1224** ± 0.0025	**0.0442** ± 0.0019
% Improv.	+1.92%	+2.15%	+1.59%	+1.68%	+1.32%	+1.61%

Table 4. Overall performance on datasets.

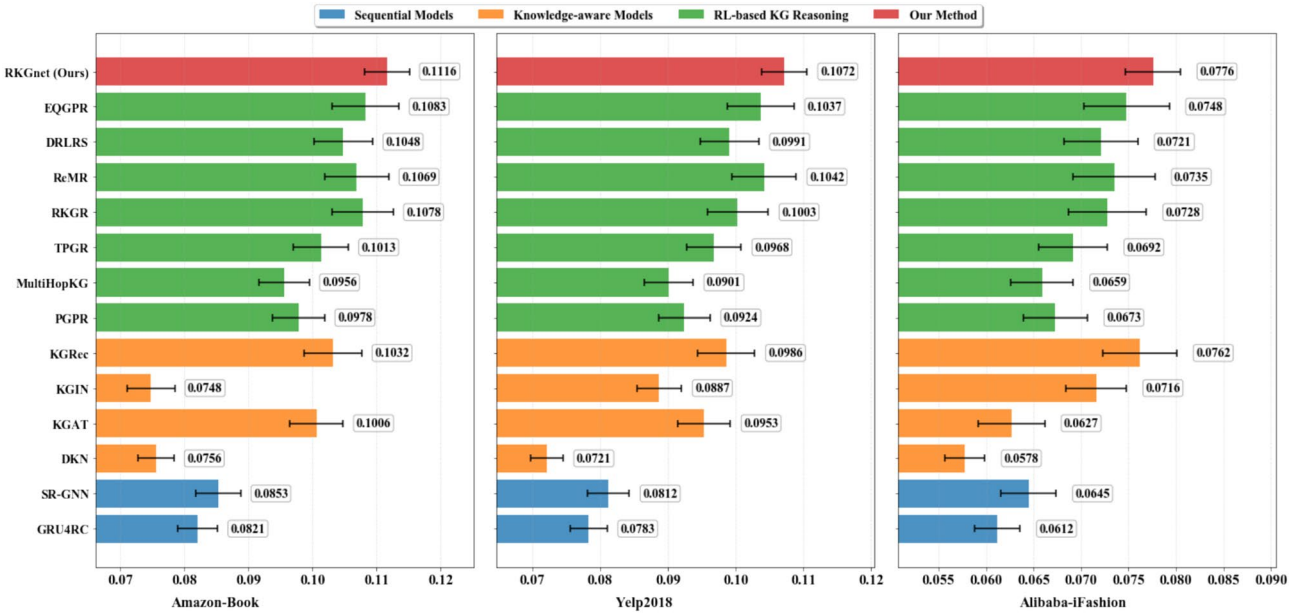

Fig. 6. NDCG@20 comparison on datasets.

into three, which include sequential models (blue bars), knowledge-aware models (orange bars), and RL-based KG reasoning models (green bars), where the proposed RKGnet (highlighted in red) performs better with all datasets than any other method. In the case of Amazon-Book and Yelp2018, RKGnet has the best Recall and Precision values, and it is significant relative to the other baseline approaches. RKGnet also works well in the Alibaba-iFashion data, outperforming the other models. The findings indicate that RKGnet, based on RL and KG-based reasoning, offers a considerable performance improvement over the traditional sequential models and other knowledge-aware models. The standard deviations represented in the error bars of the figure show how the results are consistent when the experiment is repeated.

Table 4, and Fig. 6 show a detailed comparison of different recommendation methods using three datasets, such as Amazon-Book, Yelp2018, and Alibaba-iFashion. The table presents the performance of various models, encompassing traditional approaches approaches, knowledge-driven models, and RL techniques applied to KGs. The RKGnet performed better than the others in Recall and Precision at 20. RephraseRKGnet performs the best in all datasets, showing improvements of up to 2.15% in Recall and 2.15% in Precision on Amazon-Book. The chart shows that RKGnet performs better than the other models in all datasets. This highlights the benefits of combining RL with KG reasoning. The error bars in the figure indicate the reliability of the results, as the

performance improvements are similar across tests. These results show that RKGnet is good at making better recommendations, especially when dealing with different types of data.

Ablation study

We systematically evaluate the contribution of each key component in RKGnet by removing or modifying it and measuring the impact on recommendation performance. In ablation study, we evaluate five variants of the RKGnet framework from both model components and strategy perspectives; *w/o KG*: Removes KG integration, relying solely on user-item interactions. *W/o RL* denotes the replacement of the RL module with static GNN-based preference propagation. *w/o* R_{kg} eliminates the knowledge-level reward, retaining only sequence-level optimization; *w/o* R_{seq} removes the sequence-level reward, relying solely on KG-based rewards. *w/o PPO* substitutes Proximal Policy Optimization with vanilla policy gradients. In addition, we also evaluate the effect of two important strategies, these are, *w/o CG*, which removes the candidate generation (CG) mechanism, relying only on static features to generate recommendations; and *w/ EN* which introduces noise to the edges in the knowledge graph to evaluate the model's robustness under noisy graph structures. Each variant isolates the contribution of a specific component or strategy in the full RKGnet framework, which provides insights into how each design choice influences the recommendation performance.

The results of the ablation study can be seen in Table 5. Each dataset has different versions of the RKGnet model, where some parts are taken out or changed to see how it affects performance. The Recall@20, Precision@20, NDCG@20, and Latency are considered in this evaluation. The results show that the full form of the RKGnet model performs best across all datasets for Recall, Precision, and NDCG. Whenever we remove some modules, its performance decreases. For instance, removing the KG (without KG) leads to significant decreases in performance, especially in Recall and Precision. When we remove the sequence-level reward (without R_{seq}) leads to the significant drop in performance for all the datasets. This shows how important this module is. Adding edge noise (w/ EN) makes the performance a little better, especially in how quickly it responds. This shows that using noise can make the model stronger. Also, removing the candidate generator (w/o CG) causes a slight drop in performance. This means that the CG module helps improve the quality of recommendations. This ablation study shows that every module is important for how well RKGnet works, especially the use of knowledge graphs and rewards at the sequence level.

Variant	Recall@20	Precision@20	NDCG@20	Latency(ms)
Amazon-Book				
Full RKGnet	0.1592	0.0523	0.1116	1580
w/o KG	0.1465 (– 7.9%)	0.0481 (– 8.0%)	0.1027 (– 8.0%)	1420 (– 10.1%)
w/o RL	0.1518 (– 4.6%)	0.0499 (– 4.6%)	0.1064 (– 4.7%)	1450 (– 8.2%)
w/o R_{kg}	0.1560 (– 2.0%)	0.0512 (– 2.1%)	0.1093 (– 2.1%)	1550 (– 1.9%)
w/o R_{seq}	0.1305 (– 18.0%)	0.0429 (– 18.0%)	0.0915 (– 18.0%)	1480 (– 6.3%)
w/o PPO	0.1528 (– 4.0%)	0.0502 (– 4.0%)	0.1071 (– 4.0%)	1600 (+ 1.3%)
w/o CG	0.1512 (– 4.7%)	0.0500 (– 4.4%)	0.1049 (– 5.9%)	1500 (– 5.1%)
w/ EN	0.1533 (– 2.5%)	0.0515 (– 1.5%)	0.1078 (– 3.4%)	1605 (+ 1.6%)
Yelp2018				
Full RKGnet	0.1468	0.0483	0.1072	1520
w/o KG	0.1352 (– 7.9%)	0.0444 (– 8.1%)	0.0986 (– 8.0%)	1365 (– 10.2%)
w/o RL	0.1402 (– 4.5%)	0.0461 (– 4.6%)	0.1023 (– 4.6%)	1395 (– 8.2%)
w/o R_{kg}	0.1439 (– 2.0%)	0.0473 (– 2.1%)	0.1051 (– 2.0%)	1490 (– 2.0%)
w/o R_{seq}	0.1204 (– 18.0%)	0.0396 (– 18.0%)	0.0879 (– 18.0%)	1425 (-6.3%)
w/o PPO	0.1410 (– 4.0%)	0.0464 (– 3.9%)	0.1029 (– 4.0%)	1540 (+ 1.3%)
w/o CG	0.1395 (– 5.0%)	0.0452 (– 6.4%)	0.1010 (– 6.3%)	1460 (– 4.0%)
w/ EN	0.1420 (– 3.3%)	0.0469 (– 3.0%)	0.1040 (– 3.0%)	1575 (+ 2.4%)
Alibaba-iFashion				
Full RKGnet	0.1224	0.0442	0.0776	1450
w/o KG	0.1126 (– 8.0%)	0.0406 (– 8.1%)	0.0714 (– 8.0%)	1305 (– 10.0%)
w/o RL	0.1168 (– 4.6%)	0.0421 (– 4.8%)	0.0740 (– 4.6%)	1330 (– 8.3%)
w/o R_{kg}	0.1200 (– 2.0%)	0.0433 (– 2.0%)	0.0760 (– 2.1%)	1420 (– 2.1%)
w/o R_{seq}	0.1004 (– 18.0%)	0.0362 (– 18.1%)	0.0636 (– 18.0%)	1360 (– 6.2%)
w/o PPO	0.1175 (– 4.0%)	0.0424 (– 4.1%)	0.0745 (– 4.0%)	1470 (+ 1.4%)
w/o CG	0.1150 (– 6.0%)	0.0410 (– 7.3%)	0.0722 (– 6.9%)	1390 (– 4.1%)
w/ EN	0.1191 (– 2.7%)	0.0431 (– 2.5%)	0.0754 (– 2.9%)	1500 (+ 3.4%)

Table 5. Ablation study results on datasets.

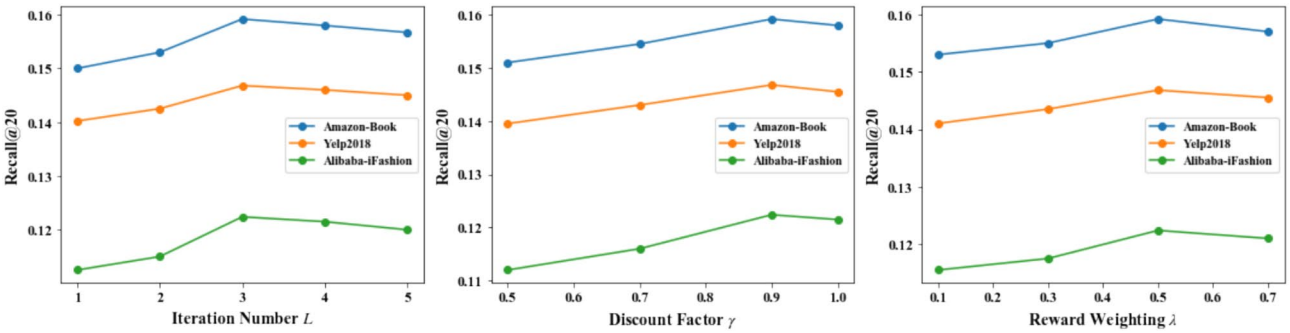

Fig. 7. Hyperparameter sensetivity analysis on datasets.

Hyperparameter sensetivity analysis

A sensitivity analysis on three important hyperparameters, such as *Iteration Number* (L), *Discount Factor* (γ), and *Reward Weighting* (λ) (R_{kg} *and* R_{seq}) are performed to comprehensively evaluate the performance of the RKGnet. These parameters play an important role in RL models, affecting convergence speed, policy optimization stability, and overall recommendation performance. By analyzing these hyperparameters, we attempt to better understand their impact on the model's performance and provide guidance for further optimization. The hyperparameter sensitivity analysis is shown in Fig. 7.

Effect of iteration number

Here, we change the iteration number L to study its impact on model refinement. A small L may lead to insufficient updates and weak preference modeling, whereas a large L allows more pairwise comparisons for better generalization but increases computation cost. This experiment shows the balance between training efficiency and recommendation performance. Increasing L generally improves Recall@20, with diminishing returns after a certain point. Optimal performance is typically achieved between 3 and 4 iterations, which balance the accuracy and computational efficiency.

Effect of discount factor

The discount factor γ governs the trade-off between immediate and long-term rewards. A lower value prioritizes short-term engagement (e.g., immediate clicks), while a higher value places greater emphasis on future returns, thereby encouraging the capture of long-term user preferences. A higher discount factor emphasizes long-term rewards, enhancing model performance in most cases. A value of γ around 0.9 yields the best results across datasets, although performance improvements plateau at higher values.

Effect of reward weighting

The parameter λ balances the knowledge graph reward and sequence reward. A higher λ emphasizes the knowledge graph's structure, while a lower λ focuses on dynamic user behavior. Proper tuning of λ improves both recommendation accuracy and interpretability. The results show that for datasets, the best recall is achieved at $\lambda = 0.5$, indicating an optimal balance between knowledge graph and sequence rewards. Extreme values of λ result in reduced performance, highlighting the importance of balancing both rewards for effective recommendations.

Discussion

Although the core concept of combining DRL with KGs has been explored in prior work, our proposed method introduces novel elements that offer tangible improvements. The distinctive contributions of the proposed method lie in its integration of path-based exploration within the KG, which allow for more dynamic and interpretable modeling of user-item interactions. In addition, the hybrid reward mechanism combine sequence-level and knowledge-based rewards, enhances the system's ability to balance both short-term and long-term objectives. These design choices are carefully tailored to address specific challenges, such as cold-start problems, and to ensure the model's adaptability and stability in real-world recommendation. Furthermore, the ablation studies provide empirical evidence that each component plays a critical role in improving the overall performance of the system. While the theoretical justification for some design choices is a direction for further exploration, the results presented in this work demonstrate clear advantages over existing baselines, validating the relevance and necessity of our proposed approach.

The current evaluation focuses on three datasets, such as the Amazon-Book, Yelp2018, and Alibaba-iFashion, we acknowledge that additional testing on a wider range of datasets is needed to fully assess the generalizability and robustness of the proposed RKGnet. Future work will aim to evaluate RKGnet on other real-world recommendation tasks, including domains such as e-commerce and social media, and to explore its performance on more diverse user behavior patterns. This will help us better understand how RKGnet adapts to varying data distributions and refine its effectiveness further.

Conclusion and future work

Recommendation systems are an important tools that help users make sense of the vast and often overwhelming amount of information they encounter in today's digital world. They are widely used across industries and research to deliver personalized content. While the integration of DL with CF has led to noticeable improvements in recommendation accuracy, key challenges like the cold-start problem and limited interpretability still persist. To overcome these problems, we introduce RKGnet, a novel recommendation framework that combines the strengths of KG with DRL. RKGnet exploits the rich relational structure of KG to model user-item interactions and uses RL to adaptively explore and update user preferences over time. By dynamically selecting and evaluating entities within the KG, the model captures multi-level, latent user interests and refines its recommendation strategy through continuous learning and feedback. In this study, only the attribute information of the item is used to build the candidate item set. The RKGnet effectively addresses the cold start problem by leveraging item metadata and KG. The results show that this approach significantly improves recommendation performance, especially for new items with limited user interaction data. The proposed approach not only boosts recommendation accuracy but also improves transparency by offering interpretable insights into why certain items are recommended. Experimental results show that RKGnet outperforms existing models in terms of accuracy, robustness, and interpretability, demonstrating its strong potential for real-world deployment in recommendation scenarios. In future work, we can add the user's information, and the relationship between users.

Data availability

All relevant data are within the paper.

Received: 26 June 2024; Accepted: 29 November 2025
Published online: 23 January 2026

References

1. Yin, N. A big data analysis method based on modified collaborative filtering recommendation algorithms. *Open Phys.* **17**, 966–974 (2019).
2. Worthy, J., Daly-Lesch, A., Tily, S., Godfrey, V. & Salmerón, C. A critical evaluation of dyslexia information on the internet. *J. Lit. Res.* **53**, 5–28 (2021).
3. Yu, X. et al. A privacy-preserving cross-domain healthcare wearables recommendation algorithm based on domain-dependent and domain-independent feature fusion. *IEEE J. Biomed. Health Inform.* **26**, 1928–1936 (2021).
4. Zheng, G., Yu, H. & Xu, W. Collaborative filtering recommendation algorithm with item label features. *Int. Core J. Eng.* **6**, 160–170 (2020).
5. Wu, L., Sun, P., Hong, R., Ge, Y. & Wang, M. Collaborative neural social recommendation. *IEEE Trans. Syst. Man Cybern. Syst.* **51**, 464–476 (2018).
6. Shakeel, N. & Shakeel, S. Context-free word importance scores for attacking neural networks. *J. Comput. Cogn. Eng.* **1**, 187–192 (2022).
7. Huang, L., Ma, Y., Liu, Y. & He, K. Dan-snr: A deep attentive network for social-aware next point-of-interest recommendation. *ACM Trans. Internet Technol.* **21**, 1–27 (2020).
8. Gu, P., Han, Y., Gao, W., Xu, G. & Wu, J. Enhancing session-based social recommendation through item graph embedding and contextual friendship modeling. *Neurocomputing* **419**, 190–202 (2021).
9. Drif, A. & Cherifi, H. Migan: Mutual-interaction graph attention network for collaborative filtering. *Entropy* **24**, 1084 (2022).
10. Drif, A., Zerrad, H. E. & Cherifi, H. Ensvae: Ensemble variational autoencoders for recommendations. *IEEE Access* **8**, 188335–188351 (2020).
11. Zhao, G., Lei, X., Qian, X. & Mei, T. Exploring users' internal influence from reviews for social recommendation. *IEEE Trans. Multimedia* **21**, 771–781 (2018).
12. Zhong, T. et al. Hybrid graph convolutional networks with multi-head attention for location recommendation. *World Wide Web* **23**, 3125–3151 (2020).
13. Ghai, S. & Trachtenberg, J. Internet information on focal prostate cancer therapy: help or hindrance? *Nat. Rev. Urol.* **16**, 337–338 (2019).
14. Rouhi Ardeshiri, R. & Ma, C. Multivariate gated recurrent unit for battery remaining useful life prediction: A deep learning approach. *Int. J. Energy Res.* **45**, 16633–16648 (2021).
15. He, X. et al. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web* 173–182 (2017).
16. Zhang, X., Zhang, J. & Yang, J. Personalized recommendation algorithm in social networks based on representation learning. *J. Intell. Fuzzy Syst.* **1**, 1–9 (2021).
17. Sun, Z., Hu, Y., Li, W., Feng, S. & Pei, L. Prediction model for short-term traffic flow based on a k-means-gated recurrent unit combination. *IET Intel. Transport Syst.* **16**, 675–690 (2022).
18. Williams, A. et al. Quality of internet information to aid patient decision making in locally advanced and recurrent rectal cancer. *Surgeon* **20**, e382–e391 (2022).
19. Liu, X. et al. Real-time poi recommendation via modeling long-and short-term user preferences. *Neurocomputing* **467**, 454–464 (2022).
20. Wang, L., Song, X. & Cong, W. Research on movie recommendation algorithm based on stack de-noising auto-encoder. *J. Phys. Conf. Ser.* **1871**, 012114 (2021).
21. Jiang, N. et al. San: Attention-based social aggregation neural networks for recommendation system. *Int. J. Intell. Syst.* **37**, 3373–3393 (2022).
22. Shen, X. & Chung, F.-L. Deep network embedding with aggregated proximity preserving. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017* 40–43 (2017).
23. Xu, H., Chai, L., Luo, Z. & Li, S. Stock movement prediction via gated recurrent unit network based on reinforcement learning with incorporated attention mechanisms. *Neurocomputing* **467**, 214–228 (2022).
24. Nerurkar, P., Chandane, M. & Bhirud, S. Survey of network embedding techniques for social networks. *Turk. J. Electr. Eng. Comput. Sci.* **27**, 4768–4782 (2019).
25. Li, H., Diao, X., Cao, J., Zhang, L. & Feng, Q. Tag-aware recommendation based on bayesian personalized ranking and feature mapping. *Intell. Data Anal.* **23**, 641–659 (2019).
26. Wang, H. et al. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* 417–426 (2018).
27. Wang, X. et al. Reinforced negative sampling over knowledge graph for recommendation. *Proc. Web Conf.* **2020**, 99–109 (2020).

28. Ranzato, M., Chopra, S., Auli, M. & Zaremba, W. Sequence level training with recurrent neural networks. Preprint at http://arxiv.org/abs/1511.06732 (2015).
29. Wang, P. et al. Kerl: A knowledge-guided reinforcement learning model for sequential recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* 209–218 (2020).
30. Zhao, W. X. et al. Kb4rec: A dataset for linking knowledge bases with recommender systems. In *Proceedings of the CIKM 2018 Workshops Co-located with 27th ACM International Conference on Information and Knowledge Management (CIKM 2018), Torino, Italy, October 22, 2018, vol. 2482 of CEUR Workshop Proceedings* (eds. Cuzzocrea, A. et al.) (CEUR-WS.org, 2018).
31. Wang, X., He, X., Cao, Y., Liu, M. & Chua, T.-S. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* 950–958 (2019).
32. Cho, K. et al. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (eds. Moschitti, A. et al.) 1724–1734. https://doi.org/10.3115/v1/D14-1179 (Association for Computational Linguistics, 2014).
33. Wu, S. et al. Session-based recommendation with graph neural networks. *Proc. AAAI Conf. Artif. Intell.* **33**, 346–353 (2019).
34. Wang, H., Zhang, F., Xie, X. & Guo, M. Dkn: Deep knowledge-aware network for news recommendation. In *Proceedings of the 2018 World Wide Web Conference* 1835–1844 (2018).
35. Wang, X. et al. Learning intents behind interactions with knowledge graph for recommendation. *Proc. Web Conf.* **2021**, 878–887 (2021).
36. Yang, Y., Huang, C., Xia, L. & Huang, C. Knowledge graph self-supervised rationalization for recommendation. In *KDD 2023, Long Beach, CA, USA, August 6–10, 2023* (eds. Singh, A. K. et al.) 3046–3056. https://doi.org/10.1145/3580305.3599400 (ACM, 2023).
37. Xian, Y., Fu, Z., Muthukrishnan, S., De Melo, G. & Zhang, Y. Reinforcement knowledge graph reasoning for explainable recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* 285–294 (2019).
38. Chen, H. et al. Large-scale interactive recommendation with tree-structured policy gradient. *Proc. AAAI Conf. Artif. Intell.* **33**, 3312–3320 (2019).
39. Zhang, S. et al. Reinforcement learning-based recommendation with user reviews on knowledge graphs. In *Knowledge Science, Engineering and Management—16th International Conference, KSEM 2023, Guangzhou, China, August 16–18, 2023, Proceedings, Part III, vol. 14119 of Lecture Notes in Computer Science* (eds. Jin, Z. et al.) 148–159. https://doi.org/10.1007/978-3-031-40289-0_12 (Springer, 2023).
40. Wang, X. et al. Multi-level recommendation reasoning over knowledge graphs with reinforcement learning. In *Proceedings of the ACM Web Conference 2022, WWW '22* 2098–2108. https://doi.org/10.1145/3485447.3512083 (Association for Computing Machinery, 2022).
41. Zhang, P., Yuan, Y., Guo, L. & Liu, H. Near-optimal control for time-varying linear discrete systems with additive nonlinearities and random gains. *IEEE Trans. Autom. Control* **64**, 2968–2975 (2018).
42. Balloccu, G., Boratto, L., Fenu, G. & Marras, M. Reinforcement recommendation reasoning through knowledge graphs for explanation path quality. *Knowl. Based Syst.* **260**, 110098 (2023).

Acknowledgements

This work is supported by the Higher Education Science Research Project of the Education Department of Ningxia Hui Autonomous Region (NGY2022143).

Author contributions

J.L.Z contributed to the study conception, methodology design, experimental investigation, and initial draft writing. Y.G and Y.L.Z were jointly responsible for method validation and data analysis; D.R.S oversaw the research guidance and manuscript revision; H.J.Z completed the data processing. All authors have read and endorsed the final manuscript.

Declarations

Competing interests

The authors declare no competing interests.

Additional information

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

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.