Applied Soft Computing 131 (2022) 109745

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Self-Supervised Reinforcement Learning with dual-reward for knowledge-aware recommendation

Wei Zhang [a,1], Yuanguo Lin [a,1], Yong Liu [b], Huanyu You [c], Pengcheng Wu [b], Fan Lin [a,*], Xiuze Zhou [d,*]

[a] School of Informatics, Xiamen University, Xiamen, China
[b] Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly (LILY), Nanyang Technological University, Singapore
[c] School of Electrical and Computer Engineering, Xiamen University Malaysia, Selangor, Malaysia
[d] Shuye Tech., Hangzhou, China

ARTICLE INFO

Article history:
Received 6 May 2022
Received in revised form 19 September 2022
Accepted 15 October 2022
Available online 26 October 2022

Keywords:
Reinforcement learning
Self-Supervised
Recommendation
Knowledge graph
Dual-reward

ABSTRACT

To improve the recommendation accuracy and offer explanations for recommendations, Reinforcement Learning (RL) has been applied to path reasoning over knowledge graphs. However, in recommendation tasks, most existing RL methods learn the path-finding policy using only a short-term or single reward, leading to a local optimum and losing some potential paths. To address these issues, we propose a Self-Supervised Reinforcement Learning (SSRL) framework combined with dual-reward for knowledge-aware recommendation reasoning over knowledge graphs. Then, we improve Actor–Critic algorithm by using a dual-reward driven strategy, which combines short-term reward with long-term incremental evaluation. The improved algorithm helps the policy guide path reasoning in an overall situation. In addition, to find the most potential paths, in the improved Actor–Critic algorithm, a loss constraint of each sample is used as a reinforced signal to update the gradients. With some improvements against baselines, experimental results demonstrate the effectiveness of our framework.

1. Introduction

Explainable recommendations aim to improve the effectiveness, transparency, and trustworthiness of recommender systems [1–3]. Knowledge reasoning over Knowledge Graph (KG) learns relations from large-scale data by reducing errors of illogical random walk [4]. Therefore, recently, in recommender systems, KG-based explainable recommendation has attracted much attention [5–9]. KG-based methods for explainable recommendations are classified into two kinds: embedding-based and path-based. Embedding-based methods (*e.g.*, TransH [10] and TransR [11]) model entities and relations by the representation distance. Path-based methods [7,12] perform path reasoning during path-finding, and Gao et al. [13] proposed the concept of meta-paths to reason over KGs. However, the approach has difficulty in dealing with multiple types of relationships and entities in large real-world KGs, and thus cannot explore the relationships between disconnected entities. Most methods have a problem in that they are unable to perform explicit reasoning due to the lack of some assisted methods to generate an effective path search policy.

An effective solution is to apply Reinforcement Learning (RL) [14,15] to perform path reasoning over a KG, where the agent guides the path-finding in a Markov Decision Process (MDP) environment. Different from ontological reasoning from rules [16,17], RL which is heuristic and unsupervised, is possible to produce more diverse results. Also, in finding paths of KG, RL learns discriminative degrees of the rules [18]. In recent years, some explainable recommendation methods combined with RL and KG have been proposed. For example, Wang et al. [19], proposed to enhance state representations with KG information considering both exploitation and exploration and designed a composite reward function to compute both sequences- and knowledge-level rewards. Furthermore, to achieve faster convergence and better interpretability, Zhao et al. [20] applied an Adversarial Actor–Critic (ADAC) model with expert path demonstrations to find interpretable reasoning paths.

However, the existing RL-based recommendation models fail to find the most potential paths, since they learn the path-finding policy according to the short-term or single reward. For instance, Xian et al. [6], following a multi-hop scoring function and soft reward strategy, first proposed the Policy-Guided Path Reasoning

* Corresponding authors.
E-mail addresses: wzhang18@stu.xmu.edu.cn (W. Zhang), xdlyg@stu.xmu.edu.cn (Y. Lin), stephenliu@ntu.edu.sg (Y. Liu), dmt1909212@xmu.edu.my (H. You), pengcheng.wu@ntu.edu.sg (P. Wu), iamafan@xmu.edu.cn (F. Lin), zhouxiuze@foxmail.com (X. Zhou).
[1] Co-first authors.

https://doi.org/10.1016/j.asoc.2022.109745
1568-4946/

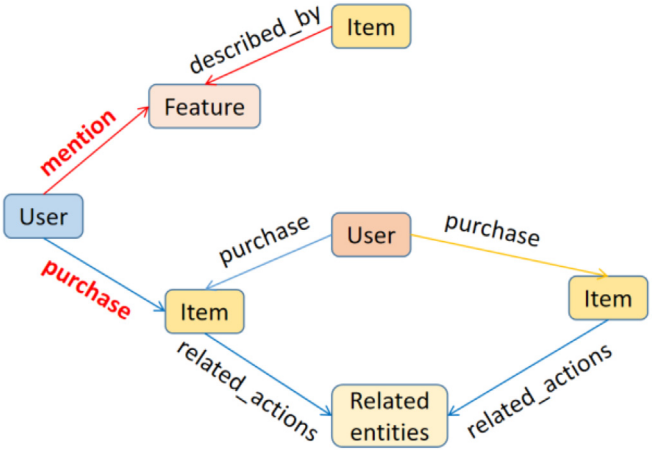


Fig. 1. Part of Valid path patterns in the results. *Path 1* (red color): *User* $\xrightarrow{mention}$ *Feature* $\xleftarrow{described\_by}$ *Item* $\xrightarrow{noop}$ *Item*. *Path 2* (blue color): *User* $\xrightarrow{purchase}$ *Item* $\xrightarrow{related\_actions}$ *Related_entities* $\xleftarrow{related\_actions}$ *Item*. Existing models generally prefer to search *Path 1* by short-term or single reward. Our method aims to transfer part of *Path 1* to *Path 2* to provide a convincing explanation.

(PGPR) model to sample reasoning paths for personalized recommendations. However, PGPR has only one reward function and lacks auxiliary rewards to drive the strategy to search for some potential paths, which leads to the missing of some effective paths when generating interpretable paths.

An example of an e-commerce recommendation scenario is illustrated in Fig. 1. In Fig. 1, a user has two main paths: mention and purchase. Most existing models may obtain more rewards when they find the shorter path of the "mention" relation. However, when searching for the path that starts from the "mention" relation, these models ignore the evaluation ability of policy networks. Specifically, in the training process, the policy is updated according to the terminal reward, whereas the potential evaluation ability of policy networks is not carried out. Hence, in the process of reasoning, if the policy valuation of the state is higher than the true value, these models do not learn path-finding policy effectively and reduce the recommendation accuracy.

To address the above issues, we propose a Self-Supervised Reinforcement Learning (SSRL) framework, which performs knowledge-aware recommendation reasoning over KGs. More specifically, we propose an improved Actor–Critic algorithm with a dual-reward driven strategy, which uses short-term reward as the policy to search for relevant paths and uses long-term incremental evaluation to infer future multi-hop paths with more convincing explanations. In addition, to find and recommend the most potential path for users, a reinforced loss constraint for each sample is introduced as a self-supervised signal combined with RL loss to jointly train the recommendation reasoning. In this way, the SSRL framework provides convincing explanations.

The main contributions of this paper are summarized as follows:

(1) We propose a SSRL framework to automatically guide the recommendation reasoning over KGs.

(2) We propose a dual-reward driven strategy for the Actor–Critic algorithm, which combines short-term and long-term knowledge-aware to perform path reasoning.

(3) We train the recommendation reasoning over KGs in a self-supervised way, which combines a reinforced loss constraint and RL loss for gradient updates.

2. Related work

2.1. Reinforcement learning

RL algorithms, learning optimal behaviors via trial-and-error interactions with an environment [21], have been applied to various fields. For instance, recurrent RL is suitable to solve the problems of equity trading [22] and investment decision making [23]. Besides, some promising RL-based models were used in different recommendation scenarios, including conversational recommendations [24,25], sequential recommendations [26–28], and explainable recommendations [3,29]. In addition, Deep Reinforcement Learning (DRL) [30], such as Trust Region Policy Optimization (TRPO) [31], Deep Q-Networks (DQN) [32], and Actor–Critic algorithms [33], leverages deep learning methods to develop RL for many different tasks.

The RL algorithms are classified into three groups for the ways of acting, that is, value-function methods (*e.g.*, Q-Learning [34]), policy search approaches (*e.g.*, REINFORCE algorithm [35]), and Actor–Critic algorithms (*e.g.*, Asynchronous Advantage Actor–Critic [36]). The value-function methods employ the maximal value to learn the optimal policy indirectly. These methods use sampling strategies, leading to slow convergence [37,38].

In contrast to value-function methods, the policy-search approaches optimize the policy directly. In particular, the policy gradient algorithm [35] improves the performance by tuning the policy parameter. However, the policy-search approaches require a lot of samples to ensure convergence.

Actor–Critic algorithm fully utilizes the advantages of policy search (actor) and value-function (critic) approaches: actor-network updates the policy gradient of the value function in terms of a critic's feedback, and critic-network uses Temporal Difference (TD) learning to update the value function in one step. For example, Liu et al. [39] proposed an interactive recommendation framework based on Actor–Critic algorithm, in which the actor-network generates the recommendation scores for the recommended items, and the critic network estimates a state–action value, combining a novel state representation with the generated action by the actor-network. Besides, Yu et al. [40] proposed a recommendation tracker to track the users' preferences based on a history of multi-modal matching rewards. The policy is updated via the Actor–Critic algorithm, to recommend the items with desired attributes to the users. The Actor–Critic algorithms will address the issue of sampling efficiency with the experience replay. Nevertheless, they often suffer from stability when performing the value evaluation and the policy update together.

To improve the Actor–Critic algorithm, researchers designed many advanced models, including Soft Actor–Critic (SAC) [41], Deep Deterministic Policy Gradient (DDPG) [42], and Shared Experience Actor–Critic (SEAC) [43]. Differing from most existing Actor–Critic algorithms, we improve the Actor–Critic algorithm in two aspects: (i) sequentially add through long-term incremental evaluation; (ii) parameterize by both reinforced loss constraint and RL loss in a self-supervised manner.

2.2. Knowledge graphs for recommendation

Recommender systems aim to help users to find what they want from a huge number of items [44–47]. KG is an effective tool for alleviating the problems of cold-start and data sparsity. Also, KG-based methods have other important advantages in that they make accurate and explainable recommendations. Existing KG-based recommendation models are classified into two categories: embedding-based and path-based approaches.

Embedding-based approaches, including TransE [48], TransH [10], TransR [11], and TransD [49], learn entities and relations

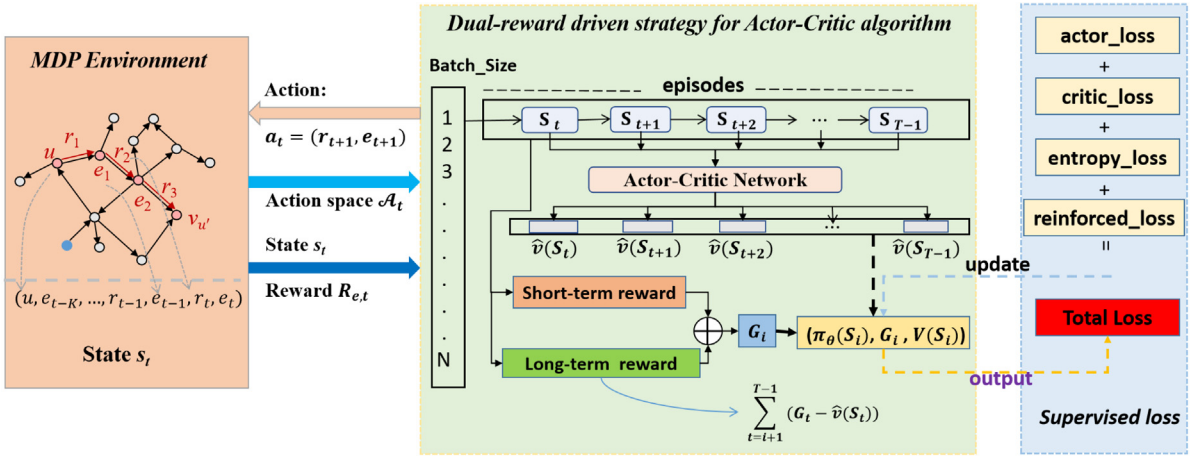


Fig. 2. SSRL framework for knowledge-aware recommendation reasoning.

in the KG and give related explanations of recommendations. Path-based approaches, which use the connection patterns in KGs to predict the link, are divided into two kinds: one adopts the meta-structure (*e.g.*, meta-path [50,51]) to learn the users' preferences from similar users/items; the other encodes connection patterns between item–item or user–item pairs into vectors to achieve better recommendation performance [52]. Nevertheless, it is impossible to search for all paths in large-scale KGs for recommendations. To tackle this problem, RL algorithms are preferred solutions.

Differing from the recommendation models based on traditional machine learning methods [53–55] and deep learning methods [56–59], the explainable recommendation models with KG and RL [60] not only make high-quality recommendations but also provide explanations, which contribute to the effectiveness and trustworthiness of the recommender systems. For example, Park et al. [61] introduced a Sentiment-Aware Knowledge Graph (SAKG) to generate convincing explanations with sentiment analysis, and proposed a sentiment-aware policy learning method to make recommendations and perform the reasoning over the SAKG. Besides, PGPR, an explainable recommendation learning framework based on knowledge map and meta path, uses the Actor–Critic algorithm to recommend actual paths in a KG [6]. In addition, to identify interpretable reasoning paths in the KG, ADAC model combines the Generative Adversarial Networks (GAN) and Actor–Critic algorithm [20].

The main challenge of the existing recommendation models with Actor–Critic algorithms is that they only consider the short-term reward in the current state and ignore the influence of the action selected according to the Temporal Difference increment in the subsequent state. Thus, such recommendation models may produce some pseudo-optimal paths, which not only reduces the recommendation accuracy but also weakens the persuasion in explaining the recommended path-finding. To solve this challenge, we improve the Actor–Critic algorithm by using a dual-reward driven strategy, which combines both short-term and long-term knowledge-aware to perform path reasoning over the KG, and thereby improves the recommendation accuracy and explainability.

3. Proposed method

3.1. SSRL framework

As shown in Fig. 2, the SSRL framework consists of two main parts: a Markov Decision Process (MDP) environment of the KG and an improved Actor–Critic network. In recommendation tasks, the actor-network is conducted to learn a recommendation path-finding policy π_Θ, on the action space $\mathcal{A}_t$, in the state S_t; the critic network evaluates the value of the state and generates the estimates of state-value function $\hat{v}(S_t)$.

In this paper, we propose an improved Actor–Critic algorithm with a dual-reward driven strategy to optimize the SSRL framework. The algorithm, using a short-term reward to find the relevant path and a long-term incremental evaluation to infer the future multi-hop path, generates more convincing explanations. First, the initial actor-network with the defined meta-path generates a finite recommended path. In a finite MDP, the current state S_t, is fed to the critic network to generate the return G_t, and the evaluation value $\hat{v}(S_t)$, of the current state. Next, the Temporal Difference increment $(G_t - \hat{v}(S_t))$, in the current state is calculated. Finally, the Temporal Difference increments of all subsequent states are accumulated as a long-term incremental evaluation. In this way, the dual-reward driven strategy, combining the short-term reward with the long-term incremental evaluation, encourages the policy to make better recommendations. During the model training, to find the most potential path-finding, we also introduce a reinforced loss constraint to supervise the reasoning.

3.2. Markov decision process environment

3.2.1. Knowledge graph

Generally, a special type of KG for the explainable recommendation $\mathcal{G}_\mathcal{R}$, with relation set $\mathcal{R}$, is defined as follows:

$$G = \{(e_{head}, r, e_{tail}) \mid e_{head}, e_{tail} \in E, r \in \mathcal{R}\}, \tag{1}$$

where the triplet (e_{head}, r, e_{tail}) denotes the relation r, from the head entity e_{head}, to the tail entity e_{tail}. $\mathcal{G}_\mathcal{R}$ contains a series of user entities U, and a series of item entities I. Both U and I belong to entity set E. We use relation r, to build the relationship between these two entities. In this paper, the relation of user u, and item i, is defined as $r_{u,i} \in R$. Then, multi-hop paths are generated to link these entities and build relationships in the knowledge map. According to prior knowledge, meta paths for path-finding, which are defined as a series of triple sets, correspond to a meta explanation strategy.

3.2.2. Markov decision process

Markov Decision Process (MDP, as a deterministic MDP) contains five basic elements: state, action, state transition probability, discount factor, and reward.

State. At time t, the state S_t is defined as a triple (u, e_t, h_t), where u denotes a user in U; e_t represents the entity that the

agent arrives after step t; and h_t represents the history before time step t. We define the n-step history as the path sequence of entities and relationships included in the past n-step, *i.e.*, $\{e_{t-n}, r_{t-n+1}, \ldots, e_{t-1}, r_t\}$. Based on a user u, the initialization state is defined as follows:

$$S_0 = (u, u, \emptyset), \tag{2}$$

where $\emptyset$ denotes the empty set at the initialization state.

Action. In a KG, for the state s_t, entity e_t will execute an action a_t to reach the next entity e_{t+1}. $a_t = (e_t, r_{t+1}) \in A_t$, where e_t denotes the current entity and r_{t+1} denotes the relationship between e_t and e_{t+1}. A set of possible actions of e_t called its action space A_t, which is defined as follows:

$$A_t = \{(r, e) \mid (e_t, r, e) \in \mathcal{G}_{\mathcal{R}}, e \notin \{e_0, \ldots, e_{t-1}\}\}. \tag{3}$$

Due to the randomness of RL strategies in the process of action selection, it is difficult for RL models to balance the diversity and accuracy of recommendation results. To alleviate this problem, inspired by the solution of PGPR [6], we introduce a behavior pruning strategy. To effectively retain the edge with high correlation, the strategy gives the score of the entity correlation through a multi-hop scoring function under the condition of the given initial user u. The strategy considers that the higher the score of the scoring function on an edge, the more likely the edge should be selected.

State transition probability. In the Markov decision-making process, given are the current state, $S_t = (u, e_t, h_t)$. After performing action $a_t = (r_{t+1}, e_{t+1})$, the agent reaches the next state. The state transition probability is defined as follows:

$$\mathbb{P}\left[S_{t+1} = (u, e_{t+1}, h_{t+1}) \mid S_t = (u, e_t, h_t), a_t = (r_{t+1}, e_{t+1})\right] \tag{4}$$

Discount factor. To attain more rewards, the agent often considers the immediate reward and future timely rewards. In a given period of MDP, the total return, G_t, is defined as follows:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{T-t-1} R_T, \tag{5}$$

where T denotes the termination status, and the discount factor, γ, belongs to [0,1], *i.e.*, the superposition of the timely current reward and the discount value of the future reward.

Reward. In the PGPR, given the multi-hop scoring function, $f(u, i)$, a soft reward mechanism is used to calculate the terminal reward R_T:

$$R_T = \begin{cases} \max\left(0, \frac{f(u,e_T)}{\max_{i \in I} f(u,i)}\right), & \text{if } e_T \in I \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where R_T is normalized to the range of $[0, 1]$. A short-term or single reward may fall into local optimality. Thus, to address this issue, we use a long-term incremental evaluation arising from the integration of Temporal Difference (TD) increments in various future states, which is presented in the following section.

3.3. Multi-hop scoring function

A multi-hop scoring function is used to prune the action space and calculate the value of the reward function. In the definition of multi-hop scoring function, we allow reverse edges in the paths of KG.

Now we define a multi-hop scoring function $f\left(e_0, e_k \mid \widetilde{r}_{k,i}\right)$ as follows.

$$f\left(e_0, e_k \mid \widetilde{r}_{k,i}\right) = \left\langle \mathbf{e}_0 + \sum_{s=1}^{i} \mathbf{r}_s, \mathbf{e_k} + \sum_{s=i+1}^{k} \mathbf{r}_s \right\rangle + b_{e_k} \tag{7}$$

where $\widetilde{r}_{k,i}$ belongs to a 1-reverse k-hop pattern, it can be presented as form of $e_0 \xrightarrow{r_1} \cdots \xrightarrow{r_i} e_i \xleftarrow{r_{i+1}} e_{i+1} \xleftarrow{r_{i+2}} \cdots \xleftarrow{r_k} e_k$. $\langle \cdot, \cdot \rangle$ is the dot product operation. e is the embedding vector of entity and r is the embedding vector of relation, $b_e k$ is the bias for entity e. When k and i are assigned different values, $\widetilde{r}_{k,i}$ represents different meanings.

When $k = 0, i = 0$, the $f\left(e_0, e_k \mid \widetilde{r}_{k,i}\right)$ represents the similarity between two vectors.

When $k = 1, i = 1$, the $f\left(e_0, e_k \mid \widetilde{r}_{k,i}\right)$ is the distance from the head vector plus the relation vector to the tail vector. We can use it to calculate timely rewards and action space pruning. When calculating the timely rewards, for a directly connected entity pair(e_i, e_{i+1}), if the relationship between the two entities changes, the distance between the sum of the head entity node and the relationship edge and the tail entity node will also change, so the associated edge between the nodes is closely related to the calculated reward value.

For $k \geq 1, 1 \leq i \leq k$, the $f\left(e_0, e_k \mid \widetilde{r}_{k,i}\right)$ is the similarity of two entities based on a 1-reverse k-hop pattern. We can use it to calculate terminal rewards. It can be seen from $f\left(e_0, e_k \mid \widetilde{r}_{k,i}\right)$ that the calculation of terminal rewards is related to the source node, the terminal node and a series of relational edges in the process, and these actions cover all the relationship types mentioned in Fig. 6. When some relational edges change in the process, the value of $f\left(e_0, e_k \mid \widetilde{r}_{k,i}\right)$ will change.

3.4. Actor–critic networks

Actor network. This network aim to learn a path-finding policy, π_θ, to calculate the probability distribution of each selected action, a_t, in the current state, S_t, $(\pi_\theta(S_t))$. The input of the actor-network is the action space and the state of the current node. The output is the probability distribution of each action in the action space. Next, the mask operation is used to delete an invalid action; then the result is fed into *softmax* to generate the final action probability distribution.

Critic network. The critic network is used to calculate the value of critic neural network in the state, S_t. In the critic network, the input is the current state, S_t, and the output is the value evaluation of the state, $\hat{v}(S_t)$.

3.4.1. Dual-reward driven strategy

Policy updating for most existing Actor–Critic based models considers only the current state of the Temporal Difference increment and ignores the impact of the future state of the Temporal Difference increment. The short-term reward, G_t, is a reward without the feedback of Temporal Difference increment in each future state. Actor–Critic networks, trained only by G_t, ignore the impact of the current state selection on the subsequent state and the corresponding Temporal Difference increment and easily fall into local optimum, leading the agent to possibly choose the suboptimal action at the beginning of path-finding.

To help the policy guide path reasoning in an overall situation, we propose an improved Actor–Critic algorithm. As shown in Fig. 3, *Dual_R*: a dual-reward driven strategy, combining the short-term reward, G_t, with long-term incremental evaluation, TD_{LT}. TD_{LT} considers the feedback of Temporal Difference increment in each future state under the policy, π_θ. *Dual_R* is defined as follows:

$$Dual\_R = G_t + TD_{LT} \tag{8}$$

$$TD_{LT} = \sum_{i=t+1}^{T-1} \left(G_i - \hat{v}(S_i)\right). \tag{9}$$

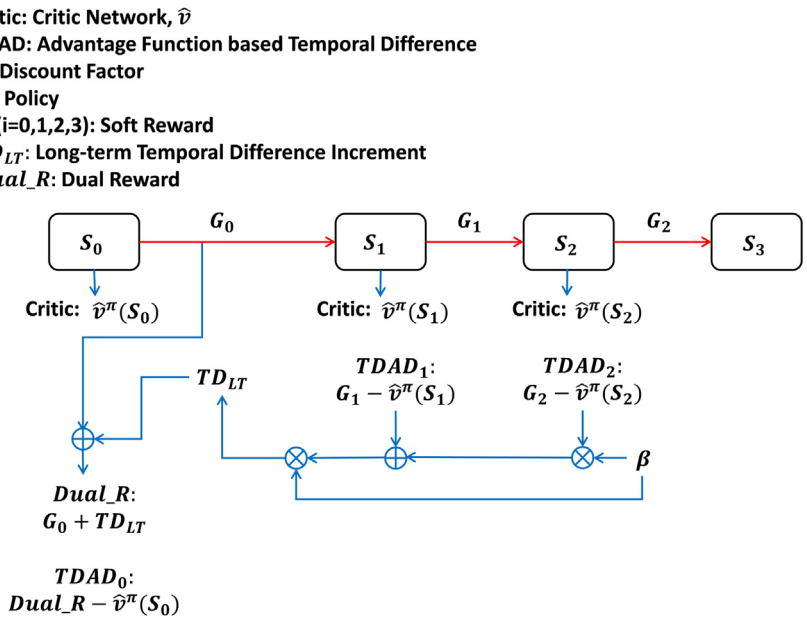


Fig. 3. Advantage value of S_0 ($State_0$) based on Dual_Reward.

The object function is defined as follows:

$$\nabla_{\Theta} J(\Theta) = \mathbb{E}_{\pi}\left[\nabla_{\Theta} \log \pi_{\Theta}(S)\,(Dual\_R - \hat{v}(S))\right], \tag{10}$$

where the difference between $Dual\_R$ and $\hat{v}(S)$ is defined as the advantage value of the corresponding state in RL. The advantage value is used to express the D-value between the real reward value and the expected reward value obtained in the state. If the cumulative attenuation of the $Dual\_R$ value in the future state is greater than the value of $\hat{v}(S)$, it indicates that the action selection strategy corresponding to the current state is relatively high quality and can be maintained. Instead, the policy needs to be updated. Thus, the advantage function value generated based on the dual-reward mechanism can effectively reflect the influence of current action selection on the future state trend.

In recommendation or path reasoning, the policy of dual-reward driven strategy tries to ensure that the Temporal Difference increment of the current and future states is positive after selecting the action, thereby recommending the most potential items to users and generating more convincing reasoning paths.

3.4.2. Reinforced loss constraint

To ensure the exploratory strategy of RL training, our model has three loss functions: critic network, actor-network, and entropy.

For the critic network, the input is the state, S_t, and the output is the expected value of the corresponding income of the state, $\hat{v}(S_t)$. The loss function is defined as follows:

$$\mathrm{L}_{critic} = \sum_{t=0}^{T-1} \left(Dual\_R - \hat{v}(S_t)\right)^2 \tag{11}$$

For the actor-network, given S_t, generates the model strategy, $\pi_{\theta}(S_t)$, which means selecting the probability value corresponding to an action. To update the parameters of actor-network adaptively, we add reinforced loss, L_r, as a constraint to the network. The loss function is defined as follows:

$$L_r = \frac{\sum_{i=0}^{batch\_size-1} e^{-\log(\pi_{\theta}(S_t)) * \left(G_t - \hat{v}(S_t)\right)[i]-1}}{batch\_size}, \tag{12}$$

$$L_{actor} = -\log(\pi_{\theta}(S_t)) * \left(Dual\_R - \hat{v}(S_t)\right) + L_r, \tag{13}$$

where L_{actor} consists of two parts: the object function of action value and L_r. The former, based on the double reward mechanism, is used to obtain the maximum value expectation generated by the current action selection. And the reinforced loss L_r is used to calculate the difference between the value evaluation of critic-network and the value generated by the action. When the value evaluation of critic-network is lower than the actual value of the action, it means that the action selection will bring more value than expected, and the L_r will drive the policy to choose such actions, on the contrary, it will reduce the selection of such actions. Therefore, the L_r can dynamically adjust the action selection policy of the Actor-network.

When training the RL network, a loss function of entropy, $\mathrm{H}(\pi)$, is added to ensure that the search strategy of the agent is exploratory as well as exploitative, so that the agent tries to select some new actions. The loss function is defined as:

$$\mathrm{L}_{entropy} = \max_{\pi_{\theta}} \mathrm{H}(\pi) \tag{14}$$

Then, the loss function of our model is defined as follows:

$$\mathrm{L} = L_{\mathrm{actor}} + L_{\mathrm{critic}} + \beta * L_{\mathrm{entropy}} \tag{15}$$

where β denotes a parameter to control the relative contribution of the entropy loss.

Finally, the policy gradient $\nabla_{\theta} J(\theta)$ is defined as follows:

$$\nabla_{\theta} J(\theta) = E_{\pi}\left[\nabla_{\theta} \log \pi_{\theta}(S_t)\,(Dual\_R - \hat{v}(S_t))\right] \tag{16}$$

3.5. Model training

Our SSRL trains the model by the improved Actor–Critic algorithm using a dual reward-driven strategy and the reinforced loss constraint as a self-supervised signal. First, we initialize the parameters of Actor–Critic network, define the meta path, and create the virtual environment of RL. Then, a recommended path with an episode length, T, is generated under the predefined meta path. The state, S_t, at time t is taken as the input of the actor and critic networks. The output of the actor-network is the evaluation, $\hat{v}(S_t)$, *i.e.*, the expected value of the corresponding action sampling in state S_t. After learning the recommended path with episode length, T, we calculate the cumulative reward, G_t,

Table 1
Statistics of two Amazon e-commerce data sets: Clothing and Beauty.

Data set	Number of users	Number of items	Relation types	Entity types	Number of product per user
Clothing	39 387	23 033	8	5	7.08
Beauty	22 363	12 101	8	5	8.88

Algorithm 1 SSRL Framework Training Process

Input: actor-network: π_θ (**s**), critic network: $\hat{v}(\ s)$, states: S, the learning rate: Lr, epoch: N, batch size: K, meta path, entity embedding, relation embedding.
Output: π_θ
initialize Actor–Critic network parameters, virtual environment of RL.
$n \leftarrow 0$
for n to N **do**
generate a episode by $\pi_\theta(\mathbf{s})$:$S_0, A_0, R_1, S_1, A_1, R_2, \cdots\cdots S_{T-1}, A_{T-1}, R_T$
$G_t \leftarrow 0$
for $t \leftarrow 0$ to $(T-1)$ **do**
for $i \leftarrow (T-1)$ to t **do**
$G_t \leftarrow \gamma G_t + R_t$
$TD_t = G_t - \hat{v}(\ S_t)$
end for
$Dual\_R = \sum_{t+1}^{T-1} TD_t + G_t$
$\hat{v}(S_t) \leftarrow \hat{v}(S_t) + \alpha\left[Dual\_R - \hat{v}(S_t)\right]$
$L_{critic} = \sum_{t=0}^{T-1}\left(Dual\_R - \hat{v}(S_t)\right)^2$
$L_r = \frac{\sum_{i=0}^{batch\_size-1} e^{-\log(\pi_\theta(S_t))*(G_t-\hat{v}(S_t))[i]-1}}{batch\_size}$
$L_{actor} = -\log\left(\pi_\theta\ (S)\right) * \left(Dual\_R - \hat{v}(S_t)\right) + L_r$
$L_{\text{entropy}} = -H(\pi_\theta)$
$\text{Total}_{loss} = L_{\text{actor}} + L_{\text{critic}} + \beta * L_{\text{entropy}}$
end for
$\nabla_\theta J(\theta) = E_\pi\left[\nabla_\theta \log \pi_\theta\ (S)\,(Dual\_R - \hat{v}(S))\right]$
end for
return π_θ

from each state to the end state and the corresponding Temporal Difference increment $(G_t - \hat{v}(S_t))$ for each state. In this way, we design a new reward as follows: a long-term incremental evaluation based on the cumulative increment of Temporal Difference in the future state relative to the current state followed by combining the short-term reward, G_T, to update the policy. Next, we update the loss function including critic loss, actor loss, entropy loss, and reinforced loss. Finally, the policy gradient, $\nabla_\theta J(\theta)$, is updated, The training time of RL policy is about six hours, and the training is shown in Algorithm 1.

4. Experiments

4.1. Experiment setup

4.1.1. Data sets

We performed experiments on two real-world e-commerce data sets: Amazon_beauty and Amazon_cloth, which were collected from Amazon and available from Amazon Review Data.[2] Both data sets consist of product reviews, meta information, and links. Each data set contains eight types of relations and five types of entities. The statistics about the data sets are shown in Table 1. The descriptions of data types are shown in Table 2. In our experiments, the data set is divided into training and testing sets in a ratio of 3:1.

[2] https://nijianmo.github.io/amazon/.

4.1.2. Baseline methods

We compare SSRL with the following recommendation methods:

BPR [53] is a Bayesian pair-wise model to learn user interests. Compared with BPR which only uses user–item interactions, SSRL uses lots of auxiliary information. The results show the necessity of auxiliary features in recommendations.

VBPR [54], visual Bayesian personalized ranking, combines BPR and visual product knowledge. Compared with VBPR which uses a little auxiliary information, SSRL uses a lot of auxiliary information to show the necessity of the number of auxiliary features.

TransRec [56] first uses translation-based embedding for the sequential recommendation and then maps user and project representations to the shared embedded space. Both TransRec and SSRL use the Embedding method, and the comparison of both methods shows the effectiveness of RL or auxiliary information.

DeepCoNN [57] encodes users and products by using a convolutional recommendation model for reviews. Both DeepCoNN and SSRL use product review information, and the comparison of both methods shows the effectiveness of RL or the amount of auxiliary information.

CKE [55] learns user interests by integrating matrix factorization and heterogeneous data. Both CKE and SSRL use abundant auxiliary information, and the comparison for both methods indicates the effectiveness of RL combined with KG.

JRL [58], the state-of-the-art joint representation learning model, uses multimodal information including image, text, and rating

Table 2
The descriptions of data types of two Amazon e-commerce data sets: Clothing and Beauty.

Entities	Description
User	User in recommender system
Item	Product to be recommended to users
Feature	A product feature word from reviews
Brand	Brand or manufacturer of the product
Category	Category of the product
Relations	**Description**
Purchase	$User \xrightarrow{purchase} Item$
Mention	$User \xrightarrow{mention} Feature$
Described_by	$Item \xrightarrow{described\_by} Feature$
Belong_to	$Item \xrightarrow{belong\_to} Category$
Produced_by	$Item \xrightarrow{produced\_by} Brand$
Also_bought	$Item \xrightarrow{also\_bought} Item$
Also_viewed	$Item \xrightarrow{also\_viewed} another\ Item$
Bought_together	$Item \xrightarrow{bought\_together} another\ Item$

to apply to the neural network. JRL using multi-modal information representation achieves the accuracy results showing the performance of RL.

LightGCN [59], a popular recommendation model based on GNNs, extends NGCF by removing feature transformation and nonlinear activation, and achieves a trade-off between the performance and efficiency. LightGCN is selected as a comparison algorithm to show coverage.

PGPR [6], an explainable recommendation learning framework based on knowledge map and meta path, uses Actor–Critic networks to find and recommend paths. Both PGPR and SSRL are unsupervised explainable recommendation algorithms, and their results reflect the effect of interpretability.

4.1.3. Evaluation metrics

Evaluation metrics: Precision, Recall, Normalized Discounted Cumulative Gain (NDCG), and Hit Ratio (HR) of top-10, are used to evaluate recommendation performance. If hit_u indicates the number of the right items in the list, all metrics are defined as follows:

$$Precision@K = \frac{1}{m}\sum_{u=1}^{m}\frac{hit_u}{K}, \tag{17}$$

$$Recall@K = \frac{1}{m}\sum_{u=1}^{m}\frac{hit_u}{N_u}, \tag{18}$$

$$NDCG@K = \frac{1}{IDCG@K}\sum_{i=1}^{K}\frac{2^{rel_i}-1}{log_2\,(i+1)}, \tag{19}$$

$$HR@K = \frac{\sum_{u=1}^{m} hit_u}{\sum_{u=1}^{m} N_u}. \tag{20}$$

where m denotes the number of users; N_u denotes the number of items rated by user u; IDCG@K denotes the Top-K list of the best recommendation results for a user; rel_i denotes the graded relevance of the item ranked at position i in the recommendations list.

4.1.4. Implementation details

The MDP environment construction of our model mainly refers to the PGPR framework [6]. We leverage the policy of RL based on Actor–Critic network to guide agents to search paths on KG. For the presentations of all entities and relations of KG, the embedding size is set at 100. We train the embedding based on the TransE model, the training epoch of embedding is set at 30. In RL training, the maximum size of action space for any state is set at 250. We prune the action space based on the multi-hop scoring function, and the rate of action pruning rate is set at 0.5. The cumulative discount factor of reward is set at 0.99. The actor–critic network is a two-layer network, the sizes of two-layer are 512 and 256, respectively. The loss function consists of L_r, L_{critic}, L_{actor} and $L_{entropy}$; Xavier initialization [62] is applied for our networkss; the training epoch is set at 50; the optimization function is Adam; the learning rate is 0.0001; the batch size is 32. In the validation experiment, we used the experimental results of top-10 as the comparison of accuracy and top-5 as the explainable experimental comparison.

This experiment mainly leverages RL to achieve explainable recommendations on KG constructed from Amazon data sets. The experiment consists of three parts: representation learning, RL training, and validation. In the representation learning module, TransE model is applied to learn the embeddings of eight relations and five entities in the Amazon data sets, which is applied to the training of RL and the calculation of reward in downstream tasks. In the RL training phase, we train the RL framework based on an actor–critic network, in which the value of the rewards is calculated by the multi-hop scoring function. We use the dual-reward mechanism to drive RL policy updates. After training, we use the trained policy to search for the target items on the KG, and recommend the items that enable the RL to obtain the most rewards to users as the items to be purchased.

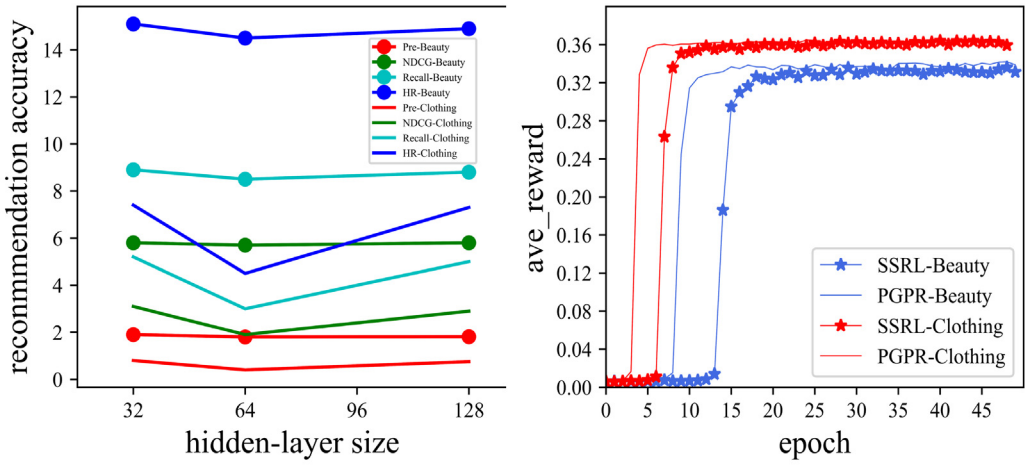
4.2. Performance comparison of recommendation

The performance of all models on two Amazon data sets is summarized in Table 3. On both data sets, the recommendation performance of SSRL is better than other baselines in terms of NDCG, HR, Precision, and Recall. For example, in terms of Recall, SSRL outperforms PGPR by 6.77% and 4.23% on the Clothing and Beauty data sets, respectively. It shows that both the quality of the policy and the recommendation accuracy is improved. The main reason is that our model uses the dual-reward and the loss constraint as the reinforced signal to update the gradient, which is conducive to the recommendation performance.

Moreover, in Fig. 4(b), the proposed SSRL framework and the PGPR framework were conducted on two Amazon data sets to show the average reward and the corresponding recommendation accuracy of each epoch. It is seen that the average reward of PGPR under the subsequent epoch is higher than that of SSRL proposed in this paper. However, in terms of accuracy, SSRL is better than PGPR. The inconsistency between ave_reward and recommendation accuracy reflects that the policy and value function of PGPR have a pseudo-high reward when they choose actions and evaluate path reward. If the agent relies on this pseudo-high reward driven strategy for action selection, it will result in a state of high reward and low accuracy. Compared with PGPR, the SSRL improves the quality of the policy by considering the feedback of the current policy selection action based on the value evaluation of the subsequent states under the current state and by using the loss constraint to correct the value function with error in the state value evaluation. The policy of SSRL does not naturally have a pseudo-high value of the total reward obtained from the generated path. Thus, the average reward per epoch of SSRL is lower than that of PGPR.

Table 3
Comparison accuracy on two data sets. The results are reported in percentage (%). The Imp. (%) indicates the percentage of accuracy improvement.

Models	Clothing				Beauty			
	Recall	Precision	HR	NDCG	Recall	Precision	HR	NDCG
BPR	1.035	0.176	1.751	0.611	4.198	1.133	8.253	2.771
VBPR	0.971	0.165	1.555	0.558	2.779	0.898	5.940	1.899
TransRec	2.071	0.309	3.109	1.236	4.851	1.279	0.863	3.215
DeepCoNN	2.328	0.230	3.291	1.314	5.426	1.118	9.798	3.361
CKE	2.501	0.386	4.281	1.498	5.941	1.373	11.032	3.699
JRL	2.979	0.439	4.630	1.731	6.942	1.543	12.769	4.393
LightGCN	3.467	0.536	5.377	2.011	7.134	1.622	13.063	4.988
PGPR	4.769	0.723	6.957	2.853	8.434	1.737	14.553	5.547
SSRL-DR	5.060	0.759	7.291	2.984	8.554	1.757	14.749	5.621
SSRL-L	4.964	0.729	7.147	2.932	8.539	1.767	14.760	5.613
SSRL	**5.092**	**0.764**	**7.367**	**3.008**	**8.791**	**1.802**	**15.035**	**5.753**
Imp. (%)	+6.77	+5.67	+5.89	+5.43	+4.23	+3.74	+3.31	+3.71



(a) Beauty and Clothing (b) Beauty and Clothing

Fig. 4. (a): Recommendation accuracy of SSRL and PGPR vs. hidden-layer sizes. (b): Average reward of SSRL and PGPR vs. epochs.

4.3. Ablation study

We conducted some experiments to verify the impact of each innovation in the proposed framework SSRL on recommendation performance with two variants of the SSRL framework:

SSRL-DR: The dual-reward driven strategy is used. The loss constraint is not used as the reinforced signal to supervise the policy gradient update.

SSRL-L: The loss constraint is used as the reinforced signal to supervise the policy gradient update. The dual-reward driven strategy is ignored.

As shown in Table 3, the recommendation accuracy of SSRL-DR and SSRL-L is better than that of PGPR, which shows that the dual-reward driven strategy and the loss constraint as a reinforcement signal to supervise the policy gradient update method improve the quality of the model's policy, thereby effectively improving the recommendation accuracy.

4.4. Parameter sensitivity study

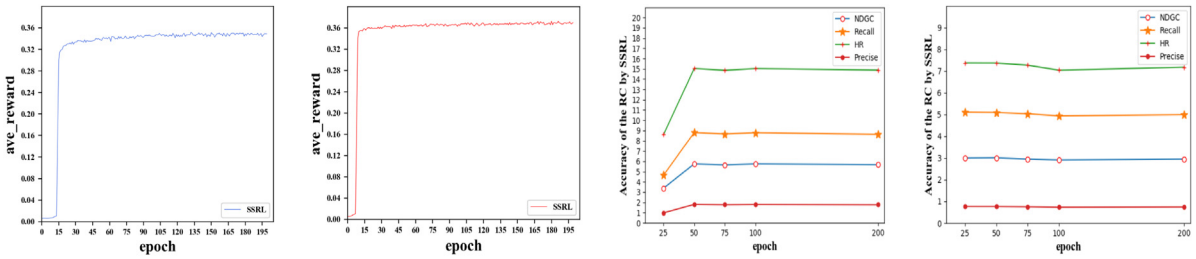
To study the impact of the two hyper-parameters of SSRL: the epoch of training and the hidden layer size of the Actor–Critic network, on the recommendation accuracy. Fig. 5 shows the accuracy of SSRL with increasing training epochs on the two Amazon data sets.

As shown in Fig. 5(a) and (c), on the Beauty data set, the average reward of the agent and the accuracy of the model begin to stabilize when the epoch is about 40. A similar trend is observed in the Clothing data set as shown in Fig. 5(b) and (d). The reason is that the model is still under-fitting in the early stage with the small training epoch, which leads to the low average reward of the agent and the low accuracy of the model. In the later stage, the model fitting is completed gradually. The effect of the model tends to be stable. In Fig. 4(a), the model performs the worst when the hidden layer size is 64, especially on the Clothing data set. The reason is that when the size is 64, the model is over-fitting on the smaller data set (Beauty), but still has room to improve on the large data sets (Clothing).

4.5. Performance comparison of explanation

Explainable recommendation generates a reasonable and effective process reasoning path when making recommendations. Table 4 compares the explainability of different explainable recommendation models, in terms of Precision and Recall, on two Amazon data sets. Baselines, SSRL-DR and SSRL-L, are proposed in this work, as well as the PGPR model.

The results show that SSRL outperforms the baselines in terms of Precision and Recall, on the Beauty and Clothing data sets. For example, compared with PGPR, SSRL improves 2.5% in terms of Recall and 2.4% in terms of Precision on Beauty; it improves 4.2% in terms of Recall and 1.5% in terms of Precision on Clothing. This indicates that the dual-reward can guide the policy to find effective recommendation paths.



(a) Amazon Beauty (b) Amazon Clothing (c) Amazon Beauty (d) Amazon Clothing

Fig. 5. Recommendation performance and average reward of SSRL vs. epochs.

Table 4
Explainability on two data sets. The percentage represents the explainability of the top-5 of the results (%). The Imp. (%) indicates the percentage of accuracy improvement.

Data set	Beauty		Clothing	
Metrics	Recall	Precision	Recall	Precision
PGPR	4.165	11.917	4.698	13.354
SSRL-DR	4.215	12.015	4.754	13.411
SSRL-L	4.20	11.956	4.746	13.399
SSRL	**4.27**	**12.204**	**4.899**	**13.56**
Imp. (%)	**2.5**	**2.4**	**4.2**	**1.5**

4.6. Effectiveness of dual-reward

This section evaluates the effectiveness of the dual-reward in the recommendation reasoning task. We show the policy optimization of the RL action selection by the dual-reward driven mechanism, which effectively adjusts the number and proportion of inference paths. In principle, the reasoning path driven with dual-reward can obtain more rewards than that of the real situation, which is also consistent with the goal of RL.

Table 5 shows the distribution of different reasoning paths distinguished by first-order and second-order actions. The reasoning paths are generated by PGPR and SSRL, respectively. It is clear that the proportion of reasoning paths with first-order action of "purchase" generated by SSRL increases, with the corresponding number of 2528 and 25 185, respectively, whereas the proportion of reasoning paths with first-order action of "mention" decreases, compared with PGPR on the Beauty and Clothing data sets. Besides, the proportion of second-order action of "described_by" in the reasoning paths generated by SSRL is higher than that generated by PGPR. The main reason is that the SSRL model makes the decision not only based on the correlation of actions in the action space in the current state but also combines with the impact of the action selections in the future state.

4.7. Case study

Understanding how to use the reasoning paths to explain the recommendation results, we give some case studies on the recommendation results with the reasoning paths, which are generated by our SSRL.

Path patterns: As shown in Fig. 1, there are some path patterns, which are abstracted from the reasoning paths generated by our model. The first-order relationship of path patterns mainly includes two types: mention and purchase. Coincidentally, the path pattern $\{User \xrightarrow{purchase} Item \xleftarrow{purchase} User \xrightarrow{purchase} Item\}$ and the path pattern $\{User \xrightarrow{purchase} Item \xrightarrow{related\_actions} Related\_entities \xleftarrow{related\_actions} Item\}$ belong to user-based collaborative filtering and content-based recommendation respectively.

Case-based paths: We provide several recommendation results with corresponding reasoning paths to demonstrate the effectiveness.

As shown in Fig. 6, from the Clothing data set, in Case 1, a user purchased an item "Owl Necklace" that was bought by "another user". Meanwhile, "another user" also purchased "Owl Head Pendant Necklace". Therefore, our model recommended "Owl Head Pendant Necklace" to this user. The reasoning path generated by Case 1 conforms to the principle of collaborative filtering, and the explainable path as the recommended result is reasonable. In Case 2, a user purchased an item "Mens 2-Bar Polo Shirt", which fell into the category "Polo Shirt". Thus, SSRL recommended to the user another item "Sport Polo Shirt" which was also in the same category "Polo Shirt". Based on the idea of collaborative filtering, the path generated by Case 2 can reasonably be used as the interpretation path of recommendations. In Case 3, a user bought an item "Apricot Scrubble Facial Wash" and also viewed another item "Eye Gel". Considering that other users who purchased "Shea Butter by Brut for Men" would also buy "Eye Gel", our method accordingly recommended "Shea Butter by Brut for Men" to the user. In Case 4, a user's reviews mentioned the feature words "Argan", so SSRL recommended the item "Argan Oil" described by "Argan" to the user. Case 4 which makes recommendations based on users' comments, accurately obtains users' needs along with some reasonable explanation paths. This is in line with the idea mentioned in ADAC [20] and PGPR [6] that the shorter the reasoning path, the more direct the relationship, and the stronger the explanation. The Case 5, a user purchased "Bracelet" and also viewed another item "Howlite Necklace". Considering that other users who purchased "Ball Stud Earrings" would also view "Howlite Necklace", SSRL accordingly recommended "Ball Stud Earrings" to the user. Also, in Case 5, the generated reasoning path conforms to the collaborative filtering idea, and can be well used as an interpretable path.

It can be seen from Table 5 that, compared with PGPR, the proposed SSRL generates more explainable paths whose first action is "purchase". Based on the results of explainable accuracy in Table 4, it shows that SSRL, to some extent, provides some explainable paths for recommendations, which cannot be supported by the paths generated by PGPR. For example, in Fig. 6, PGPR infers the Case 1 with the following reasoning path $user \xrightarrow{mention} Necklace \xleftarrow{described\_by} HowliteNecklace \xrightarrow{noop} HowliteNecklace$. However, the user in Case 1 does not plan to purchase "Howlite Necklace". From all of the above observations, we conclude that SSRL provides corresponding accurate reasoning paths for recommendations.

5. Conclusion

To solve the problem of recommendation path reasoning of RL, we proposed a SSRL framework to automatically perform the recommendation reasoning over KGs, in which an improved

Table 5
The distribution of all reasoning paths generated by PGPR and SSRL, on the Beauty and Clothing data sets.

Data set	Beauty			Clothing		
Methods	Actions(Total)	Step1 action(Ratio)	Step2 action(Ratio)	Actions(Total)	Step1 action(Ratio)	Step2 action(Ratio)
PGPR	1 329 726	mention(97.99%)	described_by(99.92%)	2 139 832	mention(91.95%)	described_by(98.4%)
			mention(0.08%)			mention(1.6%)
		purchase(2.01%)	/		purchase(7.97%)	/
SSRL	1 360 722	mention(97.84%)	described_by(99.99%)	2 341 269	mention(90.87%)	described_by(99.64%)
			mention(0.01%)			mention(0.36%)
		purchase(2.15%)	/		purchase(8.36%)	/

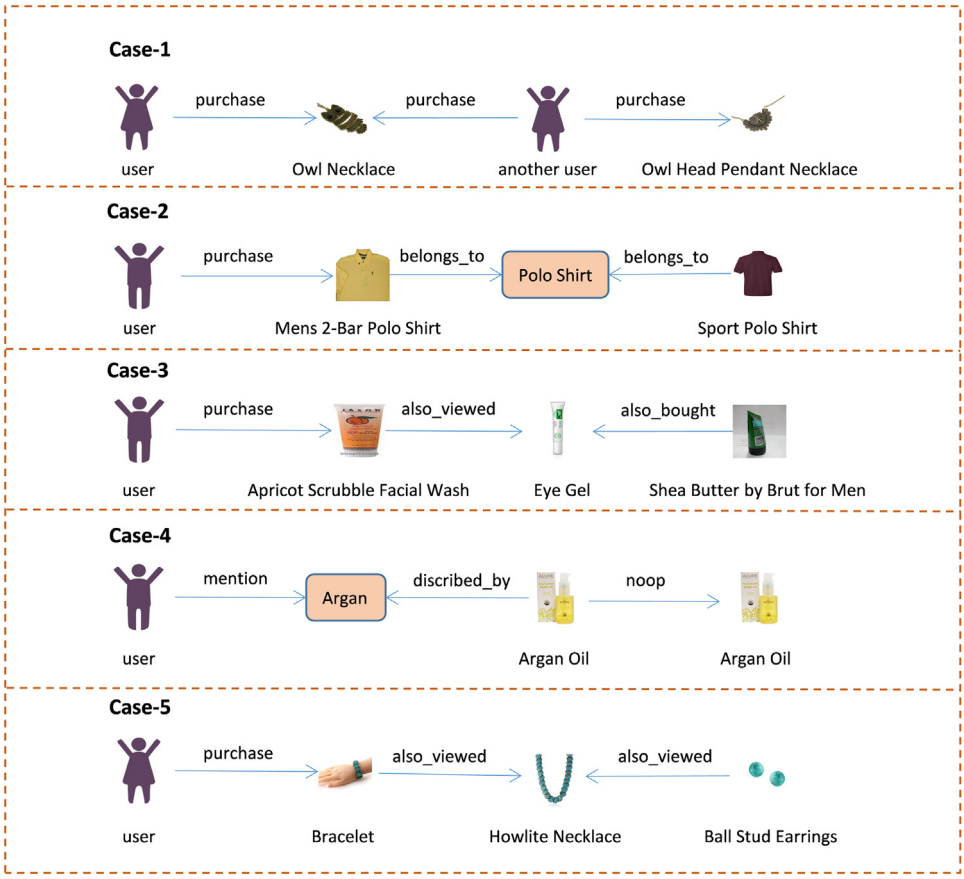


Fig. 6. Case study on recommendation reasoning task.

Actor–Critic algorithm utilizes short- and long-term knowledge-aware to perform path reasoning. To recommend the most potential path, the reinforced loss constraint was integrated into RL loss to update the gradient. Experimental results demonstrate that our proposed framework outperforms several competitive baselines. SSRL is a general framework for explainable recommendations in various fields, such as e-commerce, movie, music, and news. Our results show that the RL-based recommendation model with KG is effective with more convincing explanations. Also, the dual-reward driven strategy is more efficient in improving the recommendation performance when combining the short-term reward with the long-term incremental evaluation.

Limitations. As Actor–Critic algorithms usually cause potential information loss, the stability of SSRL which is trained by Actor–Critic algorithm, is unsatisfactory. We can alleviate this issue by using TRPO [31] to limit the step size for each gradient. Besides, like most existing RL-based recommendation methods, SSRL has not yet developed the intrinsic mechanism of RL algorithm. To better understand how SSRL generates the explanations, we will use the causal inference procedure [63] to exploit the interpretability of the SSRL model.

Future work. It would be interesting to use a Transformer method to exploit associated information of entities in KGs. Our SSRL combined with this information can further improve recommendation accuracy. Another meaningful research direction is leveraging Natural Language Processing (NLP) to generate textual sentence explanations for the reasoning paths.

CRediT authorship contribution statement

Wei Zhang: Methodology, Writing – original draft, Data curation, Software. **Yuanguo Lin:** Writing – original draft. **Yong Liu:** Investigation. **Huanyu You:** Data curation. **Pengcheng Wu:** Validation. **Fan Lin:** Project administration. **Xiuze Zhou:** Project administration, Conceptualization, Methodology.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This research is supported by the National Natural Science Foundation of China No. 61977055.

The authors would like to thank Michael McAllister for proofreading this manuscript.

References

[1] Y. Zhang, X. Chen, Explainable recommendation: A survey and new perspectives, Found. Trends Inf. Retr. 14 (1) (2020) 1–101.
[2] W. Ma, M. Zhang, Y. Cao, W. Jin, C. Wang, Y. Liu, S. Ma, X. Ren, Jointly learning explainable rules for recommendation with knowledge graph, in: Proceedings of the World Wide Web Conference, 2019, pp. 1210–1221.
[3] X. Wang, Y. Chen, J. Yang, L. Wu, Z. Wu, X. Xie, A reinforcement learning framework for explainable recommendation, in: Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM), IEEE, 2018, pp. 587–596.
[4] X. Chen, S. Jia, Y. Xiang, A review: Knowledge reasoning over knowledge graph, Expert Syst. Appl. 141 (2020) 112948.
[5] Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, H. Xiong, Q. He, A survey on knowledge graph-based recommender systems, IEEE Trans. Knowl. Data Eng. (2020).
[6] Y. Xian, Z. Fu, S. Muthukrishnan, G. De Melo, Y. Zhang, Reinforcement knowledge graph reasoning for explainable recommendation, in: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2019, pp. 285–294.
[7] X. Wang, D. Wang, C. Xu, X. He, Y. Cao, T.-S. Chua, Explainable reasoning over knowledge graphs for recommendation, in: Proceedings of the 33th AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 5329–5336.
[8] Z. Fu, Y. Xian, R. Gao, J. Zhao, Q. Huang, Y. Ge, S. Xu, S. Geng, C. Shah, Y. Zhang, et al., Fairness-aware explainable recommendation over knowledge graphs, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 69–78.
[9] T. Suzuki, S. Oyama, M. Kurihara, Explainable recommendation using review text and a knowledge graph, in: Proceedings of the IEEE International Conference on Big Data (Big Data), IEEE, 2019, pp. 4638–4643.
[10] Z. Wang, J. Zhang, J. Feng, Z. Chen, Knowledge graph embedding by translating on hyperplanes, in: Proceedings of the 28th AAAI Conference on Artificial Intelligence, Vol. 28, 2014.
[11] Y. Lin, Z. Liu, M. Sun, Y. Liu, X. Zhu, Learning entity and relation embeddings for knowledge graph completion, in: Proceedings of the 29th AAAI Conference on Artificial Intelligence, 2015, pp. 2181–2187.
[12] Y. Xian, Z. Fu, H. Zhao, Y. Ge, X. Chen, Q. Huang, S. Geng, Z. Qin, G. De Melo, S. Muthukrishnan, Y. Zhang, CAFE: Coarse-to-fine neural symbolic reasoning for explainable recommendation, in: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020, pp. 1645–1654.
[13] L. Gao, H. Yang, J. Wu, C. Zhou, W. Lu, Y. Hu, Recommendation with multi-source heterogeneous information, in: Proceedings of the 27th International Joint Conference on Artificial Intelligence, 2018, pp. 3378–3384.
[14] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.
[15] G. Ke, H.-L. Du, Y.-C. Chen, Cross-platform dynamic goods recommendation system based on reinforcement learning and social networks, Appl. Soft Comput. 104 (2021) 107213.
[16] S. Shishehchi, S.Y. Banihashem, N.A.M. Zin, A proposed semantic recommendation system for e-learning: A rule and ontology based e-learning recommendation system, in: 2010 International Symposium on Information Technology, Vol. 1, IEEE, 2010, pp. 1–5.
[17] L. Galárraga, C. Teflioudi, K. Hose, F.M. Suchanek, Fast rule mining in ontological knowledge bases with AMIE++, VLDB J. 24 (6) (2015) 707–730.
[18] B. Liu, L. Yao, Z. Ding, J. Xu, J. Wu, Combining ontology and reinforcement learning for zero-shot classification, Knowl.-Based Syst. 144 (2018) 42–50.
[19] P. Wang, Y. Fan, L. Xia, W.X. Zhao, J. Huang, KERL: A knowledge-guided reinforcement learning model for sequential recommendation, in: SIGIR '20: The 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020.
[20] K. Zhao, X. Wang, Y. Zhang, L. Zhao, Z. Liu, C. Xing, X. Xie, Leveraging demonstrations for reinforcement recommendation reasoning over knowledge graphs, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 239–248.
[21] L.P. Kaelbling, M.L. Littman, A.W. Moore, Reinforcement learning: A survey, J. Artificial Intelligence Res. 4 (1996) 237–285.
[22] P. Gabrielsson, U. Johansson, High-frequency equity index futures trading using recurrent reinforcement learning with candlesticks, in: IEEE Symposium Series on Computational Intelligence, 2015, pp. 734–741.
[23] S. Almahdi, S.Y. Yang, An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown, Expert Syst. Appl. 87 (2017) 267–279.
[24] C. Greco, A. Suglia, P. Basile, G. Semeraro, Converse-et-impera: Exploiting deep learning and hierarchical reinforcement learning for conversational recommender systems, in: Proceedings of the Conference of the Italian Association for Artificial Intelligence, Springer, 2017, pp. 372–386.
[25] M.S. Llorente, S.E. Guerrero, Increasing retrieval quality in conversational recommenders, IEEE Trans. Knowl. Data Eng. 24 (10) (2011) 1876–1888.
[26] X. Xin, A. Karatzoglou, I. Arapakis, J.M. Jose, Self-supervised reinforcement learning for recommender systems, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 931–940.
[27] O. Moling, L. Baltrunas, F. Ricci, Optimal radio channel recommendations with explicit and implicit feedback, in: Proceedings of the 6th ACM Conference on Recommender Systems, 2012, pp. 75–82.
[28] P. Wang, Y. Fan, L. Xia, W.X. Zhao, S. Niu, J. Huang, 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, 2020, pp. 209–218.
[29] Y. Lu, R. Dong, B. Smyth, Why I like it: multi-task learning for recommendation and explanation, in: Proceedings of the 12th ACM Conference on Recommender Systems, 2018, pp. 4–12.
[30] K. Arulkumaran, M.P. Deisenroth, M. Brundage, A.A. Bharath, Deep reinforcement learning: A brief survey, IEEE Signal Process. Mag. 34 (6) (2017) 26–38.
[31] J. Schulman, S. Levine, P. Abbeel, M. Jordan, P. Moritz, Trust region policy optimization, in: Proceedings of the International Conference on Machine Learning, PMLR, 2015, pp. 1889–1897.
[32] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing atari with deep reinforcement learning, 2013, arXiv preprint arXiv:1312.5602.
[33] V.R. Konda, J.N. Tsitsiklis, Actor-critic algorithms, in: Advances in Neural Information Processing Systems, 2000, pp. 1008–1014.
[34] C.J. Watkins, P. Dayan, Technical note Q-learning, Mach. Learn. 8 (3) (1992) 279–292.
[35] R.J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, Mach. Learn. 8 (3) (1992) 229–256.
[36] V. Mnih, A.P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning, in: Proceedings of the International Conference on Machine Learning, PMLR, 2016, pp. 1928–1937.
[37] G. Weisz, P. Budzianowski, P.-H. Su, M. Gašić, Sample efficient deep reinforcement learning for dialogue systems with large action spaces, IEEE/ACM Trans. Audio Speech Lang. Process. 26 (11) (2018) 2083–2097.
[38] T. Xu, Y. Liang, Sample complexity bounds for two timescale value-based reinforcement learning algorithms, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2021, pp. 811–819.
[39] F. Liu, R. Tang, X. Li, W. Zhang, Y. Ye, H. Chen, H. Guo, Y. Zhang, X. He, State representation modeling for deep reinforcement learning based recommendation, Knowl.-Based Syst. 205 (2020) 106170.
[40] T. Yu, Y. Shen, R. Zhang, X. Zeng, H. Jin, Vision-language recommendation via attribute augmented multimodal reinforcement learning, in: Proceedings of the 27th ACM International Conference on Multimedia, 2019, pp. 39–47.
[41] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: Proceedings of the International Conference on Machine Learning, PMLR, 2018, pp. 1861–1870.
[42] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, 2015, arXiv preprint arXiv:1509.02971.
[43] F. Christianos, L. Schäfer, S.V. Albrecht, Shared experience actor-critic for multi-agent reinforcement learning, in: Proceedings of the 34th Conference on Neural Information Processing Systems, Curran Associates Inc, 2020, pp. 10707–10717.

[44] M. Chen, T. Ma, X. Zhou, CoCNN: Co-occurrence CNN for recommendation, Expert Syst. Appl. 195 (2022) 116595.
[45] R. Wang, Y. Jiang, J. Lou, Attention-based dynamic user preference modeling and nonlinear feature interaction learning for collaborative filtering recommendation, Appl. Soft Comput. 110 (2021) 107652.
[46] M. Chen, Y. Li, X. Zhou, CoNet: Co-occurrence neural networks for recommendation, Future Gener. Comput. Syst. 124 (2021) 308–314.
[47] M. Chen, X. Zhou, DeepRank: Learning to rank with neural networks for recommendation, Knowl.-Based Syst. 209 (2020) 106478.
[48] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, Adv. Neural Inf. Process. Syst. 26 (2013).
[49] G. Ji, S. He, L. Xu, K. Liu, J. Zhao, Knowledge graph embedding via dynamic mapping matrix, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, Vol. 1, 2015, pp. 687–696.
[50] C. Luo, W. Pang, Z. Wang, C. Lin, Hete-cf: Social-based collaborative filtering recommendation using heterogeneous relations, in: Proceedings of the 2014 IEEE International Conference on Data Mining, IEEE, 2014, pp. 917–922.
[51] X. Yu, X. Ren, Y. Sun, B. Sturt, U. Khandelwal, Q. Gu, B. Norick, J. Han, Recommendation in heterogeneous information networks with implicit user feedback, in: Proceedings of the 7th ACM Conference on Recommender Systems, 2013, pp. 347–350.
[52] Z. Sun, J. Yang, J. Zhang, A. Bozzon, L.-K. Huang, C. Xu, Recurrent knowledge graph embedding for effective recommendation, in: Proceedings of the 12th ACM Conference on Recommender Systems, 2018, pp. 297–305.
[53] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidtthieme, BPR: Bayesian personalized ranking from implicit feedback, in: Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence, 2012, pp. 452–461.
[54] R. He, J. McAuley, VBPR: visual bayesian personalized ranking from implicit feedback, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 30, 2016.
[55] W. Zhang, Q. Yuan, J. Han, J. Wang, Collaborative multi-level embedding learning from reviews for rating prediction, in: Proceedings of the 25th International Joint Conference on Artificial Intelligence, Vol. 16, 2016, pp. 2986–2992.
[56] R. He, W.-C. Kang, J. McAuley, Translation-based recommendation, in: Proceedings of the 11th ACM Conference on Recommender Systems, 2017, pp. 161–169.
[57] L. Zheng, V. Noroozi, P.S. Yu, Joint deep modeling of users and items using reviews for recommendation, in: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, 2017, pp. 425–434.
[58] Y. Zhang, Q. Ai, X. Chen, W.B. Croft, Joint representation learning for top-n recommendation with heterogeneous information sources, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017, pp. 1449–1458.
[59] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, M. Wang, LightGCN: Simplifying and powering graph convolution network for recommendation, in: SIGIR '20: The 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020.
[60] S. Tao, R. Qiu, Y. Ping, H. Ma, Multi-modal knowledge-aware reinforcement learning network for explainable recommendation, Knowl.-Based Syst. 227 (2021) 107217.
[61] S.-J. Park, D.-K. Chae, H.-K. Bae, S. Park, S.-W. Kim, Reinforcement learning over sentiment-augmented knowledge graphs towards accurate and explainable recommendation, in: Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, 2022, pp. 784–793.
[62] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1026–1034.
[63] L. Yao, Z. Chu, S. Li, Y. Li, J. Gao, A. Zhang, A survey on causal inference, ACM Trans. Knowl. Discov. Data 15 (5) (2021) 1–46.