Reinforcement Knowledge Graph Reasoning for Explainable Recommendation

Yikun Xian[*]
Rutgers University
siriusxyk@gmail.com

Zuohui Fu[*]
Rutgers University
zuohui.fu@rutgers.edu

S. Muthukrishnan
Rutgers University
muthu@cs.rutgers.edu

Gerard de Melo
Rutgers University
gdm@demelo.org

Yongfeng Zhang
Rutgers University
yongfeng.zhang@rutgers.edu

ABSTRACT

Recent advances in personalized recommendation have sparked great interest in the exploitation of rich structured information provided by knowledge graphs. Unlike most existing approaches that only focus on leveraging knowledge graphs for more accurate recommendation, we perform explicit reasoning with knowledge for decision making so that the recommendations are generated and supported by an interpretable causal inference procedure. To this end, we propose a method called Policy-Guided Path Reasoning (PGPR), which couples recommendation and interpretability by providing actual paths in a knowledge graph. Our contributions include four aspects. We first highlight the significance of incorporating knowledge graphs into recommendation to formally define and interpret the reasoning process. Second, we propose a reinforcement learning (RL) approach featuring an innovative soft reward strategy, user-conditional action pruning and a multi-hop scoring function. Third, we design a policy-guided graph search algorithm to efficiently and effectively sample reasoning paths for recommendation. Finally, we extensively evaluate our method on several large-scale real-world benchmark datasets, obtaining favorable results compared with state-of-the-art methods.

CCS CONCEPTS

• Information systems → Collaborative filtering; **Recommender systems**; **• Computing methodologies → Machine learning**.

KEYWORDS

Recommendation System; Reinforcement Learning; Knowledge Graphs; Explainability

ACM Reference Format:
Yikun Xian, Zuohui Fu, S. Muthukrishnan, Gerard de Melo, and Yongfeng Zhang. 2019. Reinforcement Knowledge Graph Reasoning for Explainable Recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '19), July 21–25, 2019, Paris, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3331184.3331203

1 INTRODUCTION

Equipping recommendation systems with the ability to leverage knowledge graphs (KG) not only facilitates better exploitation of various structured information to improve the recommendation performance, but also enhances the explainability of recommendation models due to the intuitive ease of understanding relationships between entities [33]. Recently, researchers have explored the potential of knowledge graph reasoning in personalized recommendation. One line of research focuses on making recommendations using knowledge graph embedding models, such as TransE [2] and node2vec [5]. These approaches align the knowledge graph in a regularized vector space and uncover the similarity between entities by calculating their representation distance [30]. However, pure KG embedding methods lack the ability to discover multi-hop relational paths. Ai *et al.* [1] proposed to enhance the collaborative filtering (CF) method over KG embedding for personalized recommendation, followed by a soft matching algorithm to find explanation paths between users and items. However, one issue of this strategy is that the explanations are not produced according to the reasoning process, but instead are later generated by an empirical similarity matching between the user and item embeddings. Hence, their explanation component is merely trying to find a post-hoc explanation for the already chosen recommendations.

Another line of research investigates path-based recommendation. For example, Gao *et al.* [4] proposed the notion of meta-paths to reason over KGs. However, the approach has difficulty in coping with numerous types of relations and entities in large real-world KGs, and hence it is incapable of exploring relationships between unconnected entities. Wang *et al.* [28] first developed a path embedding approach for recommendation over KGs that enumerates all the qualified paths between every user–item pair, and then trained a sequential RNN model from the extracted paths to predict the ranking score for the pairs. The recommendation performance is further improved, but it is not practical to fully explore all the paths for each user–item pair in large-scale KGs.

We believe that an intelligent recommendation agent should have the ability to conduct explicit reasoning over knowledge graphs to make decisions, rather than merely embed the graph as latent vectors for similarity matching. In this paper, we consider knowledge graphs as a versatile structure to maintain the agent's knowledge

[*] Both authors contributed equally to this work.

SIGIR '19, July 21–25, 2019, Paris, France
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6172-9/19/07…$15.00
https://doi.org/10.1145/3331184.3331203

about users, items, other entities and their relationships. The agent starts from a user and conducts explicit multi-step path reasoning over the graph, so as to discover suitable items in the graph for recommendation to the target user. The underlying idea is that if the agent draws its conclusion based on an explicit reasoning path, it will be easy to interpret the reasoning process that leads to each recommendation. Thus, the system can provide causal evidence in support of the recommended items. Accordingly, our goal is not only to select a set of candidate items for recommendation, but also to provide the corresponding reasoning paths in the graph as interpretable evidence for why a given recommendation is made. As an example illustrated in Figure 1, given user A, the algorithm is expected to find candidate items B and F, along with their reasoning paths in the graph, e.g., $\{$User $A \longrightarrow$ Item $A \longrightarrow$ Brand $A \longrightarrow$ Item $B\}$ and $\{$User $A \longrightarrow$ Feature $B \longrightarrow$ Item $F\}$.

In this paper, we propose an approach that overcomes the shortcomings of previous work. Specifically, we cast the recommendation problem as a deterministic Markov Decision Process (MDP) over the knowledge graph. We adopt a Reinforcement Learning (RL) approach, in which an agent starts from a given user, and learns to navigate to the potential items of interest, such that the path history can serve as a genuine explanation for why the item is recommended to the user.

The main challenges are threefold. First, it is non-trivial to measure the correctness of an item for a user, so careful consideration is needed regarding the terminal conditions and RL rewards. To solve the problem, we design a soft reward strategy based on a multi-hop scoring function that leverages the rich heterogeneous information in the knowledge graph. Second, the size of the action space depends on the out-degrees in the graph, which can be very large for some nodes, so it is important to conduct an efficient exploration to find promising reasoning paths in the graph. In this regard, we propose a user-conditional action pruning strategy to decrease the size of the action spaces while guaranteeing the recommendation performance. Third, the diversity of both items and paths must be preserved when the agent is exploring the graph for recommendation, so as to avoid being trapped in limited regions of items. To achieve this, we design a policy-guided search algorithm to sample reasoning paths for recommendation in the inference phase. We conduct several case studies on the reasoning paths to qualitatively evaluate the diversity of explanations for recommendation.

The major contributions of this paper can be outlined as follows.

(1) We highlight the significance of incorporating rich heterogeneous information into the recommendation problem to formally define and interpret the reasoning process.
(2) We propose an RL-based approach to solve the problem, driven by our soft reward strategy, user-conditional action pruning, and a multi-hop scoring strategy.
(3) We design a beam search-based algorithm guided by the policy network to efficiently sample diverse reasoning paths and candidate item sets for recommendation.
(4) We extensively evaluate the effectiveness of our method on several Amazon e-commerce domains, obtaining strong results as well as explainable reasoning paths.

The source code is available online.[1]

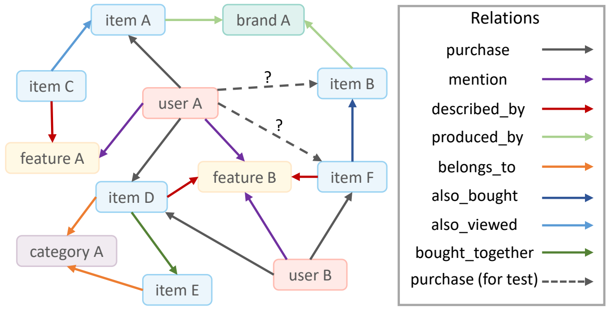


Figure 1: Illustration of the Knowledge Graph Reasoning for Explainable Recommendation (KGRE-Rec) problem.

2 RELATED WORK

2.1 Collaborative Filtering

Collaborative Filtering (CF) has been one of the most fundamental approaches for recommendation. Early approaches to CF consider the user–item rating matrix and predict ratings via user-based [11, 21] or item-based [15, 22] collaborative filtering methods. With the development of dimension reduction methods, latent factor models such as matrix factorization gained widespread adoption in recommender systems. Specific techniques include singular value decomposition [12], non-negative matrix factorization [13] and probabilistic matrix factorization [18]. For each user and item, these approaches essentially learn a latent factor representation to calculate the matching score of the user–item pairs. Recently, deep learning and neural models have further extended collaborative filtering. These are broadly classified into two sub-categories: the similarity learning approach and the representation learning approach. Similarity learning adopts fairly simple user/item embeddings (e.g., one-hot vectors) and learns a complex prediction network as a similarity function to compute user–item matching scores [9]. In contrast, the representation learning approach learns much richer user/item representations but adopts a simple similarity function (e.g., inner product) for score matching [32]. However, researchers have noticed the difficulty of explaining the recommendation results in latent factor or latent representation models, making explainable recommendation [33, 34] an important research problem for the community.

2.2 Recommendation with Knowledge Graphs

Some previous efforts have made recommendations to users with the help of knowledge graph embeddings [2, 19]. One research direction leverages knowledge graph embeddings as rich content information to enhance the recommendation performance. For example, Zhang *et al.* [30] adopted knowledge base embeddings to generate user and item representations for recommendation, while Huang *et al.* [10] employed memory networks over knowledge graph entity embeddings for recommendation. Wang *et al.* [26] proposed a ripple network approach for embedding-guided multi-hop KG-based recommendation. Another research direction attempts to leverage the entity and path information in the knowledge graph to make explainable decisions. For example, Ai *et al.* [1] incorporated the learning of knowledge graph embeddings for explainable recommendation. However, their explanation paths are essentially post-hoc explanations, as they are generated by soft matching after

[1] Link to the source code: https://github.com/orcax/PGPR

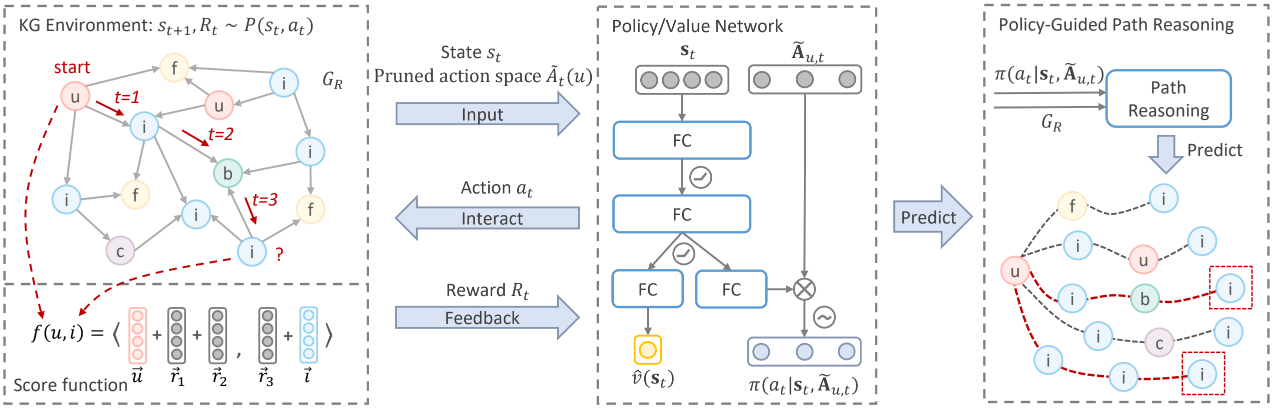

KG Environment: $s_{t+1}, R_t \sim P(s_t, a_t)$
start
G_R
u
f
i
u
i
t=1
t=2
b
i
i
f
t=3
f
i
c
i
?
$f(u,i) = \langle \vec{u} + \vec{r}_1 + \vec{r}_2 , \vec{r}_3 + \vec{i} \rangle$
Score function
State s_t
Pruned action space $\tilde{A}_t(u)$
Input
Action a_t
Interact
Reward R_t
Feedback
Policy/Value Network
$\mathbf{s}_t$
$\tilde{\mathbf{A}}_{u,t}$
FC
FC
FC
FC
$\hat{v}(\mathbf{s}_t)$
$\pi(a_t|\mathbf{s}_t, \tilde{\mathbf{A}}_{u,t})$
Predict
Policy-Guided Path Reasoning
$\pi(a_t|\mathbf{s}_t, \tilde{\mathbf{A}}_{u,t})$
G_R
Path Reasoning
Predict
u
f
i
i
u
i
i
b
i
i
c
i
i
i
i

Figure 2: Pipeline of our Policy-Guided Path Reasoning method for recommendation. The algorithm aims to learn a policy that navigates from a user to potential items of interest by interacting with the knowledge graph environment. The trained policy is then adopted for the path reasoning phase to make recommendations to the user.

the corresponding items have been chosen. Wang *et al.* [28] proposed an RNN based model to reason over KGs for recommendation. However, it requires enumerating all the possible paths between each user–item pair for model training and prediction, which can be impractical for large-scale knowledge graphs.

2.3 Reinforcement Learning

Reinforcement Learning has attracted substantial interest in the research community. In recent years, there have been a series of widely noted successful applications of deep RL approaches (e.g., AlphaGo [23]), demonstrating their ability to better understand the environment, and enabling them to infer high-level causal relationships. There have been attempts to invoke RL in recommender systems in a non-KG setting, such as for ads recommendation [25], news recommendation [35] and post-hoc explainable recommendation [27]. At the same time, researchers have also explored RL in KG settings for other tasks such as question answering (QA) [3, 14, 29], which formulates multi-hop reasoning as a sequential decision making problem. For example, Xiong *et al.* [29] leveraged reinforcement learning for path-finding, and Das *et al.* [3] proposed a system called MINERVA that trains a model for multi-hop KG question answering. Lin *et al.* [14] proposed models for end-to-end RL-based KG question answering with reward shaping and action dropout. However, to the best of our knowledge, there is no previous work utilizing RL in KGs for the task of recommendation, especially when the KG has an extremely large action space for each entity node as the number of path hops grow.

3 METHODOLOGY

In this section, we first formalize a new recommendation problem called *Knowledge Graph Reasoning for Explainable Recommendation*. Then we present our approach based on reinforcement learning over knowledge graphs to solve the problem.

3.1 Problem Formulation

In general, a knowledge graph $\mathcal{G}$ with entity set $\mathcal{E}$ and relation set $\mathcal{R}$ is defined as $\mathcal{G} = \{(e, r, e') \mid e, e' \in \mathcal{E}, r \in \mathcal{R}\}$, where each triplet (e, r, e') represents a fact of the relation r from head entity e to tail entity e'. In this paper, we consider a special type of knowledge graph for explainable recommendation, denoted by $\mathcal{G}_\mathrm{R}$. It contains a subset of a *User* entities $\mathcal{U}$ and a subset of *Item* entities $\mathcal{I}$, where $\mathcal{U}, \mathcal{I} \subseteq \mathcal{E}$ and $\mathcal{U} \cap \mathcal{I} = \varnothing$. These two kinds of entities are connected through relations r_{ui}. We give a relaxed definition of k-hop paths over the graph $\mathcal{G}_\mathrm{R}$ as follows.

Definition 3.1. (k-hop path) A k-hop path from entity e_0 to entity e_k is defined as a sequence of $k+1$ entities connected by k relations, denoted by $p_k(e_0, e_k) = \left\{ e_0 \overset{r_1}{\longleftrightarrow} e_1 \overset{r_2}{\longleftrightarrow} \cdots \overset{r_k}{\longleftrightarrow} e_k \right\}$, where $e_{i-1} \overset{r_i}{\longleftrightarrow} e_i$ represents either $(e_{i-1}, r_i, e_i) \in \mathcal{G}_\mathrm{R}$ or $(e_i, r_i, e_{i-1}) \in \mathcal{G}_\mathrm{R}$, $i \in [k]$.

Now, the problem of *Knowledge Graph Reasoning for Explainable Recommendation (KGRE-Rec)* can be formalized as below.

Definition 3.2. (KGRE-Rec Problem) Given a knowledge graph $\mathcal{G}_\mathrm{R}$, user $u \in \mathcal{U}$ and integers K and N, the goal is to find a recommendation set of items $\{i_n\}_{n \in [N]} \subseteq \mathcal{I}$ such that each pair (u, i_n) is associated with one reasoning path $p_k(u, i_n)$ $(2 \leq k \leq K)$, and N is the number of recommendations.

In order to simultaneously conduct item recommendation and path finding, we consider three aspects that result in a good solution to the problem. First, we do not have pre-defined targeted items for any user, so it is not applicable to use a binary reward indicating whether the user interacts with the item or not. A better design of the reward function is to incorporate the uncertainty of how an item is relevant to a user based on the rich heterogeneous information given by the knowledge graph. Second, out-degrees of some entities may be very large, which degrades the efficiency of finding paths from users to potential item entities. Enumeration of all possible paths between each user and all items is unfeasible on very large graphs. Thus, the key challenge is how to effectively perform edge pruning and efficiently search relevant paths towards potential items using the reward as a heuristic. Third, for every user, the diversity of reasoning paths for recommended items should be guaranteed. It is not reasonable to always stick to a specific type of reasoning path to provide explainable recommendations. One naive solution is post-hoc recommendation, which first generates candidate items according to some similarity measure, followed by a separate path finding procedure from the user to candidate items within the graph. The major downsides of this are that the recommendation process fails to leverage the rich heterogeneous meta-data in the knowledge graph, and that the generated paths are detached from the actual decision-making process adopted by the recommendation algorithm, which remains uninterpretable.

In the following sections, we introduce our *Policy-Guided Path Reasoning* method (PGPR) for explainable recommendation over knowledge graphs. It solves the problem through reinforcement learning by making recommendations while simultaneously searching for paths in the context of rich heterogeneous information in the KG. As illustrated in Figure 2, the main idea is to train an RL agent that learns to navigate to potentially "good" items conditioned on the starting user in the knowledge graph environment. The agent is then exploited to efficiently sample reasoning paths for each user leading to the recommended items. These sampled paths naturally serve as the explanations for the recommended items.

3.2 Formulation as Markov Decision Process

The starting point of our method is to formalize the KGRE-Rec problem as a Markov Decision Process (MDP) [24]. In order to guarantee path connectivity, we add two special kinds of edges to the graph $\mathcal{G}_\mathrm{R}$. The first one are reverse edges, i.e., if $(e, r, e') \in \mathcal{G}_\mathrm{R}$, then $(e', r, e) \in \mathcal{G}_\mathrm{R}$, which are used for our path definition. The second are self-loop edges, associated with the no operation (NO-OP) relation, i.e., if $e \in \mathcal{E}$, then $(e, r_\mathrm{noop}, e) \in \mathcal{G}_\mathrm{R}$.

State. The state s_t at step t is defined as a tuple (u, e_t, h_t), where $u \in \mathcal{U}$ is the starting user entity, e_t is the entity the agent has reached at step t, and h_t is the history prior to step t. We define the k-step history as the combination of all entities and relations in the past k steps, i.e., $\{e_{t-k}, r_{t-k+1}, \dots, e_{t-1}, r_t\}$. Conditioned on some user u, the initial state is represented as $s_0 = (u, u, \varnothing)$. Given some fixed horizon T, the terminal state is $s_T = (u, e_T, h_T)$.

Action. The complete action space A_t of state s_t is defined as all possible outgoing edges of entity e_t excluding history entities and relations. Formally, $A_t = \{(r, e) \mid (e_t, r, e) \in \mathcal{G}_\mathrm{R}, e \notin \{e_0, \dots, e_{t-1}\}\}$. Since the out-degree follows a long-tail distribution, some nodes have much larger out-degrees compared with the rest of nodes. It is fairly space-inefficient to maintain the size of the action space based on the largest out-degree. Thus, we introduce a *user-conditional action pruning strategy* that effectively keeps the promising edges conditioned on the starting user based on a scoring function. Specifically, the scoring function $f((r, e) \mid u)$ maps any edge (r, e) $(\forall r \in \mathcal{R}, \forall e \in \mathcal{E})$ to a real-valued score conditioned on user u. Then, the user-conditional pruned action space of state s_t, denoted by $\tilde{A}_t(u)$, is defined as:

$$\tilde{A}_t(u) = \{(r, e) \mid \mathrm{rank}(f((r, e) \mid u)) \leq \alpha, (r, e) \in A_t\}, \qquad (1)$$

where α is a pre-defined integer that upper-bounds the size of the action space. The details of this scoring function $f((r, e) \mid u)$ will be discussed in the next section.

Reward. Given any user, there is no pre-known targeted item in the KGRE-Rec problem, so it is unfeasible to consider binary rewards indicating whether the agent has reached a target or not. Instead, the agent is encouraged to explore as many "good" paths as possible. Intuitively, in the context of recommendations, a "good" path is one that leads to an item that a user will interact with, with high probability. To this end, we consider to give a soft reward only for the terminal state $s_T = (u, e_T, h_T)$ based on another scoring function $f(u, i)$. The terminal reward R_T is defined as

$$R_T = \begin{cases} \max\left(0, \frac{f(u, e_T)}{\max_{i \in \mathcal{I}} f(u, i)}\right), & \text{if } e_T \in \mathcal{I} \\ 0, & \text{otherwise,} \end{cases} \qquad (2)$$

where the value of R_T is normalized to the range of $[0, 1]$. $f(u, i)$ is also introduced in the next section.

Transition. Due to the graph properties, a state is determined by the position of the entity. Given a state $s_t = (u, e_t, h_t)$ and an action $a_t = (r_{t+1}, e_{t+1})$, the transition to the next state s_{t+1} is:

$$\mathbb{P}\left[s_{t+1} = (u, e_{t+1}, h_{t+1}) | s_t = (u, e_t, h_t), a_t = (r_{t+1}, e_{t+1})\right] = 1 \quad (3)$$

One exception is that the initial state $s_0 = (u, u, \varnothing)$ is stochastic, which is determined by the starting user entity. For simplicity, we assume the prior distribution of users follows a uniform distribution so that each user is equally sampled at the beginning.

Optimization. Based on our MDP formulation, our goal is to learn a stochastic policy π that maximizes the expected cumulative reward for any initial user u:

$$J(\theta) = \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \gamma^t R_{t+1} \,\middle|\, s_0 = (u, u, \varnothing) \right]. \qquad (4)$$

We solve the problem through *REINFORCE with baseline* [24] by designing a policy network and a value network that share the same feature layers. The policy network $\pi(\cdot|\mathbf{s}, \tilde{\mathbf{A}}_u)$ takes as input the state vector $\mathbf{s}$ and binarized vector $\tilde{\mathbf{A}}_u$ of pruned action space $\tilde{A}(u)$ and emits the probability of each action, with zero probability for actions not in $\tilde{A}(u)$. The value network $\hat{v}(\mathbf{s})$ maps the state vector $\mathbf{s}$ to a real value, which is used as the baseline in REINFORCE. The structures of the two networks are defined as follows:

$$\mathbf{x} = \mathrm{dropout}(\sigma(\mathrm{dropout}(\sigma(\mathbf{s}\mathbf{W}_1))\mathbf{W}_2)) \qquad (5)$$

$$\pi(\cdot|\mathbf{s}, \tilde{\mathbf{A}}_u) = \mathrm{softmax}(\tilde{\mathbf{A}}_u \odot (\mathbf{x}\mathbf{W}_\mathrm{p})) \qquad (6)$$

$$\hat{v}(\mathbf{s}) = \mathbf{x}\mathbf{W}_\mathrm{v} \qquad (7)$$

Here, $\mathbf{x} \in \mathbb{R}^{d_f}$ are the learned hidden features of the state, $\odot$ is the Hadamard product, which is used to mask invalid actions here, and σ is a non-linear activation function, for which we use an Exponential Linear Unit (ELU). State vectors $\mathbf{s} \in \mathbb{R}^{d_s}$ are represented as the concatenation of the embeddings u, e_t and history h_t. For the binarized pruned action space $\tilde{\mathbf{A}}_u \in \{0, 1\}^{d_A}$, we set the maximum size d_A among all pruned action spaces. The model parameters for both networks are denoted as $\Theta = \{\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_\mathrm{p}, \mathbf{W}_\mathrm{v}\}$. Additionally, we add a regularization term $H(\pi)$ that maximizes the entropy of the policy in order to encourage the agent to explore more diverse paths. Finally, the policy gradient $\nabla_\Theta J(\Theta)$ is defined as:

$$\nabla_\Theta J(\Theta) = \mathbb{E}_\pi \left[\nabla_\Theta \log \pi_\Theta(\cdot|\mathbf{s}, \tilde{\mathbf{A}}_u) \left(G - \hat{v}(\mathbf{s})\right) \right], \qquad (8)$$

where G is the discounted cumulative reward from state s to the terminal state s_T.

3.3 Multi-Hop Scoring Function

Now we present the scoring function for the action pruning strategy and the reward function. We start with some relevant concepts.

One property of the knowledge graph $\mathcal{G}_\mathrm{R}$ is that given the type of a head entity and a valid relation, the type of tail entity is determined. We can extend this property by creating a chain rule of entity and relation types: $\{e_0, r_1, e_1, r_2, \ldots, r_k, e_k\}$. If the types of entity e_0 and all relations $r_1, \ldots, r_k$ are given, the types of all other entities $e_1, \ldots, e_k$ are uniquely determined. According to this rule, we introduce the concept of *patterns* as follows.

Definition 3.3. (k-hop pattern) A sequence of k relations $\tilde{r}_k = \{r_1, \ldots, r_k\}$ is called a valid k-hop pattern for two entities (e_0, e_k) if there exists a set of entities $\{e_1, \ldots, e_{k-1}\}$ whose types are uniquely determined such that $\{e_0 \xleftrightarrow{r_1} e_1 \xleftrightarrow{r_2} \cdots \xleftrightarrow{r_{k-1}} e_{k-1} \xleftrightarrow{r_k} e_k\}$ forms a valid k-hop path over $\mathcal{G}_\mathrm{R}$.

One caveat with pattern is the direction of each relation, provided that we allow reverse edges in the path. For entities e, e' and relation r, $e \xleftrightarrow{r} e'$ represents either $e \xrightarrow{r} e'$ or $e \xleftarrow{r} e'$ in the path. We refer to the relation r as a *forward* one if $(e, r, e') \in \mathcal{G}_\mathrm{R}$ and $e \xrightarrow{r} e'$, or as a *backward* one if $(e', r, e) \in \mathcal{G}_\mathrm{R}$ and $e \xleftarrow{r} e'$.

In order to define the scoring functions for action pruning and reward, we consider a special case of patterns with both forward and backward relations.

Definition 3.4. (*1-reverse* k-hop pattern) A k-hop pattern is *1-reverse*, denoted by $\tilde{r}_{k,j} = \{r_1, \ldots, r_j, r_{j+1}, \ldots, r_k\}$ $(j \in [0, k])$, if $r_1, \ldots, r_j$ are forward and $r_{j+1} \ldots r_k$ are backward.

In other words, paths with a *1-reverse* k-hop pattern have the form of $e_0 \xrightarrow{r_1} \cdots \xrightarrow{r_j} e_j \xleftarrow{r_{j+1}} e_{j+1} \xleftarrow{r_{j+2}} \cdots \xleftarrow{r_k} e_k$. Note that the pattern contains all backward relations when $j = 0$, and all forward relations when $j = k$.

Now we define a general multi-hop scoring function $f(e_0, e_k \mid \tilde{r}_{k,j})$ of two entities e_0, e_k given *1-reverse* k-hop pattern $\widetilde{r}_{k,j}$ as follows.

$$f(e_0, e_k \mid \widetilde{r}_{k,j}) = \left\langle \mathbf{e_0} + \sum_{s=1}^{j} \mathbf{r_s},\ \mathbf{e_k} + \sum_{s=j+1}^{k} \mathbf{r_s} \right\rangle + b_{e_k}, \qquad (9)$$

where $\langle \cdot, \cdot \rangle$ is the dot product operation, $\mathbf{e}, \mathbf{r} \in \mathbb{R}^d$ are d-dimensional vector representations of the entity e and relation r, and $b_e \in \mathbb{R}$ is the bias for entity e. When $k = 0, j = 0$, the scoring function simply computes the cosine similarity between two vectors:

$$f(e_0, e_k \mid \tilde{r}_{0,0}) = \langle \mathbf{e_0}, \mathbf{e_k} \rangle + b_{e_k}. \qquad (10)$$

When $k = 1, j = 1$, the scoring function computes the similarity between two entities via translational embeddings [2]:

$$f(e_0, e_k \mid \widetilde{r}_{1,1}) = \langle \mathbf{e_0} + \mathbf{r_1},\ \mathbf{e_k} \rangle + b_{e_k} \qquad (11)$$

For $k \geq 1, 1 \leq j \leq k$, the scoring function in Equation 9 quantifies the similarity of two entities based on a 1-reverse k-hop pattern.

Scoring Function for Action Pruning. We assume that for user entity u and another entity e of other type, there exists only one *1-reverse* k-hop pattern $\tilde{r}_{k,j}$ for some integer k. For entity $e \notin \mathcal{U}$, we denote k_e as the smallest k such that $\tilde{r}_{k,j}$ is a valid pattern for entities (u, e). Therefore, the scoring function for action pruning is defined as $f((r, e) \mid u) = f(u, e \mid \tilde{r}_{k_e,j})$.

Algorithm 1: Policy-Guided Path Reasoning

```
Input  : $u$, $\pi(\cdot|\mathbf{s}, \tilde{\mathbf{A}}_u)$, $T$, $\{K_1, \ldots, K_T\}$
Output : path set $\mathcal{P}_T$, probability set $\mathcal{Q}_T$, reward set $\mathcal{R}_T$
Initialize $\mathcal{P}_0 \leftarrow \{\{u\}\}$, $\mathcal{Q}_0 \leftarrow \{1\}$, $\mathcal{R}_0 \leftarrow \{0\}$;
for $t \leftarrow 1$ to $T$ do
    Initialize $\mathcal{P}_t \leftarrow \varnothing$, $\mathcal{Q}_t \leftarrow \varnothing$, $\mathcal{R}_t \leftarrow \varnothing$;
    forall $\hat{p} \in \mathcal{P}_{t-1}$, $\hat{q} \in \mathcal{Q}_{t-1}$, $\hat{r} \in \mathcal{R}_{t-1}$ do
        ▷ path $\hat{p} \doteq \{u, r_1, \ldots, r_{t-1}, e_{t-1}\}$;
        Set $\mathbf{s}_{t-1} \leftarrow (u, e_{t-1}, h_{t-1})$;
        Get user-conditional pruned action space $\tilde{A}_{t-1}(u)$
          from environment given state $s_{t-1}$;
        ▷ $p(a) \doteq \pi(a \mid \mathbf{s}_{t-1}, \tilde{\mathbf{A}}_{u,t-1})$ and $a \doteq (r_t, e_t)$;
        Actions $\mathcal{A}_t \leftarrow \{a \mid \mathrm{rank}(p(a)) \leq K_t,\ \forall a \in \tilde{A}_{t-1}(u)\}$;
        forall $a \in \mathcal{A}_t$ do
            Get $s_t, R_{t+1}$ from environment given action $a$;
            Save new path $\hat{p} \cup \{r_t, e_t\}$ to $\mathcal{P}_t$;
            Save new probability $p(a)\,\hat{q}$ to $\mathcal{Q}_t$;
            Save new reward $R_{t+1} + \hat{r}$ to $\mathcal{R}_t$;
        end
    end
end
Save $\forall \hat{p} \in \mathcal{P}_T$ if the path $\hat{p}$ ends with an item;
return filtered $\mathcal{P}_T, \mathcal{Q}_T, \mathcal{R}_T$;
```

Scoring Function for Reward. We simply use the 1-hop pattern between user entity and item entity, i.e., $(u, r_{ui}, i) \in \mathcal{G}_\mathrm{R}$. The scoring function for reward design is defined as $f(u, i) = f(u, i|\tilde{r}_{1,1})$.

Learning Scoring Function. A natural question that arises is how to train the embeddings for each entity and relation. For any pair of entities (e, e') with valid k-hop pattern $\tilde{r}_{k,j}$, we seek to maximize the conditional probability of $\mathbb{P}(e' \mid e, \tilde{r}_{k,j})$, which is defined as

$$\mathbb{P}(e' \mid e, \tilde{r}_{k,j}) = \frac{\exp\left(f(e, e' \mid \tilde{r}_{k,j})\right)}{\sum_{e'' \in \mathcal{E}} \exp\left(f(e, e'' \mid \tilde{r}_{k,j})\right)}. \qquad (12)$$

However, due to the huge size of the entity set $\mathcal{E}$, we adopt a negative sampling technique [17] to approximate $\log \mathbb{P}(e' \mid e, \tilde{r}_{k,j})$:

$$\begin{aligned} \log \mathbb{P}(e|e', \tilde{r}_{k,j}) \approx & \log \sigma\left(f(e, e' \mid \tilde{r}_{k,j})\right) \\ & + m\mathbb{E}_{e''}\left[\log \sigma\left(-f(e, e'' \mid \tilde{r}_{k,j})\right)\right] \qquad (13) \end{aligned}$$

The goal is to maximize the objective function $J(\mathcal{G}_\mathrm{R})$, defined as:

$$J(\mathcal{G}_\mathrm{R}) = \sum_{e, e' \in \mathcal{E}} \sum_{k=1}^{K} \mathbb{1}\{(e, \tilde{r}_{k,j}, e')\} \log \mathbb{P}(e'|e, \tilde{r}_{k,j}), \qquad (14)$$

where $\mathbb{1}\{(e, \tilde{r}_{k,j}, e')\}$ is 1 if $\tilde{r}_{k,j}$ is a valid pattern for entities (e, e') and 0 otherwise.

3.4 Policy-Guided Path Reasoning

The final step is to solve our recommendation problem over the knowledge graph guided by the trained policy network. Recall that given a user u, the goal is to find a set of candidate items $\{i_n\}$ and

		CDs & Vinyl	**Clothing**	**Cell Phones**	**Beauty**
Entities	**Description**	**Number of Entities**			
User	User in recommender system	75,258	39,387	27,879	22,363
Item	Product to be recommended to users	64,443	23,033	10,429	12,101
Feature	A product feature word from reviews	202,959	21,366	22,493	22,564
Brand	Brand or manufacturer of the product	1,414	1,182	955	2,077
Category	Category of the product	770	1,193	206	248
Relations	**Description**	**Number of Relations per Head Entity**			
Purchase	*User* $\xrightarrow{\text{purchase}}$ *Item*	14.58 ± 39.13	7.08 ± 3.59	6.97 ± 4.55	8.88 ± 8.16
Mention	*User* $\xrightarrow{\text{mention}}$ *Feature*	$2,545.92 \pm 10,942.31$	440.20 ± 452.38	652.08 ± 1335.76	806.89 ± 1344.08
Described_by	*Item* $\xrightarrow{\text{described\_by}}$ *Feature*	$2,973.19 \pm 5,490.93$	752.75 ± 909.42	$1,743.16 \pm 3,482.76$	$1,491.16 \pm 2,553.93$
Belong_to	*Item* $\xrightarrow{\text{belong\_to}}$ *Category*	7.25 ± 3.13	6.72 ± 2.15	3.49 ± 1.08	4.11 ± 0.70
Produced_by	*Item* $\xrightarrow{\text{produced\_by}}$ *Brand*	0.21 ± 0.41	0.17 ± 0.38	0.52 ± 0.50	0.83 ± 0.38
Also_bought	*Item* $\xrightarrow{\text{also\_bought}}$ *Item*	57.28 ± 39.22	61.35 ± 32.99	56.53 ± 35.82	73.65 ± 30.69
Also_viewed	*Item* $\xrightarrow{\text{also\_viewed}}$ another *Item*	0.27 ± 1.86	6.29 ± 6.17	1.24 ± 4.29	12.84 ± 8.97
Bought_together	*Item* $\xrightarrow{\text{bought\_together}}$ another *Item*	0.68 ± 0.80	0.69 ± 0.90	0.81 ± 0.77	0.75 ± 0.72

Table 1: Descriptions and statistics of four Amazon e-commerce datasets: *CDs & Vinyl*, *Clothing*, *Cell Phones* and *Beauty*.

the corresponding reasoning paths $\{p_n(u, i_n)\}$. One straightforward way is to sample n paths for each user u according to the policy network $\pi(\cdot|\mathbf{s}, \tilde{\mathbf{A}}_u)$. However, this method cannot guarantee the diversity of paths, because the agent guided by the policy network is likely to repeatedly search the same path with the largest cumulative rewards. Therefore, we propose to employ beam search guided by the action probability and reward to explore the candidate paths as well as the recommended items for each user. The process is described as Algorithm 1. It takes as input the given user u, the policy network $\pi(\cdot|\mathbf{s}, \tilde{\mathbf{A}}_u)$, horizon T, and predefined sampling sizes at each step, denoted by $K_1, \ldots, K_T$. As output, it delivers a candidate set of T-hop paths $\mathcal{P}_T$ for the user with corresponding path generative probabilities $\mathcal{Q}_T$ and path rewards $\mathcal{R}_T$. Note that each path $p_T(u, i_n) \in \mathcal{P}_T$ ends with an item entity associated with a path generative probability and a path reward.

For the acquired candidate paths, there may exist multiple paths between the user u and item i_n. Thus, for each pair of (u, i_n) in the candidate set, we select the path from $\mathcal{P}_T$ with the highest generative probability based on $\mathcal{Q}_T$ as the one to interpret the reasoning process of why item i_n is recommended to u. Finally, we rank the selected interpretable paths according to the path reward in $\mathcal{R}_T$ and recommend the corresponding items to the user.

4 EXPERIMENTS

In this section, we extensively evaluate the performance of our *PGPR* method on real-world datasets. We first introduce the benchmarks for our experiments and the corresponding experimental settings. Then we quantitatively compare the effectiveness of our model with other state-of-the-art approaches, followed by ablation studies to show how parameter variations influence our model.

4.1 Data Description

All experiments are conducted on the Amazon e-commerce datasets collection [7], consisting of product reviews and meta information from Amazon.com. The datasets include four categories: *CDs and Vinyl*, *Clothing*, *Cell Phones* and *Beauty*. Each category is considered as an individual benchmark that constitutes a knowledge graph containing 5 types of entities and 7 types of relations. The description and statistics of each entity and relation can be found in Table 1. Note that once the type of head entity and relation are provided, the type of tail entity is uniquely determined. In addition, as shown in Table 1, we find that *Mention* and *Described_by* account for a very large proportion among all relations. These two relations are both connected to the *Feature* entity, which may contain redundant and less meaningful words. We thus adopt TF-IDF to eliminate less salient features in the preprocessing stage: For each dataset, we keep the frequency of feature words less than 5,000 with TF-IDF score > 0.1. We adopt the same data split rule as in previous work [32], which randomly sampled 70% of user purchases as the training data and took the rest 30% as test. The objective in the KGRE-Rec problem is to recommend items purchased by users in the test set together with reasoning paths for each user–item pair.

4.2 Experimental Setup

Baselines & Metrics. We compare our results against previous state-of-the-art methods. **BPR** [20] is a Bayesian personalized ranking model that learns latent embeddings of users and items. **BPR-HFT** [16] is a Hidden Factors and Topics (HFT) model that incorporates topic distributions to learn latent factors from reviews of users or items. **VBPR** [8] is the Visual Bayesian Personalized Ranking method that builds upon the BPR model but incorporates visual product knowledge. **TransRec** [6] invokes translation-based embeddings for sequential recommendation. It learns to map both user and item representations in a shared embedding space through personalized translation vectors. **DeepCoNN** or Deep Cooperative Neural Networks [36] are a review-based convolutional recommendation model that learns to encode both users and products with reviews assisting in rating prediction. **CKE** or Collaborative Knowledge base Embedding [31] is a modern neural recommender system based on a joint model integrating matrix factorization and heterogeneous data formats, including textual contents, visual information and a structural knowledge base to infer the top-N recommendations results. **JRL** [32] is a start-of-the-art joint representation learning model for top-N recommendation that utilizes

Dataset	**CDs & Vinyl**				**Clothing**				**Cell Phones**				**Beauty**			
Measures (%)	NDCG	Recall	HR	Prec.	NDCG	Recall	HR	Prec.	NDCG	Recall	HR	Prec.	NDCG	Recall	HR	Prec.
BPR	2.009	2.679	8.554	1.085	0.601	1.046	1.767	0.185	1.998	3.258	5.273	0.595	2.753	4.241	8.241	1.143
BPR-HFT	2.661	3.570	9.926	1.268	1.067	1.819	2.872	0.297	3.151	5.307	8.125	0.860	2.934	4.459	8.268	1.132
VBPR	0.631	0.845	2.930	0.328	0.560	0.968	1.557	0.166	1.797	3.489	5.002	0.507	1.901	2.786	5.961	0.902
TransRec	3.372	5.283	11.956	1.837	1.245	2.078	3.116	0.312	3.361	6.279	8.725	0.962	3.218	4.853	0.867	1.285
DeepCoNN	4.218	6.001	13.857	1.681	1.310	2.332	3.286	0.229	3.636	6.353	9.913	0.999	3.359	5.429	9.807	1.200
CKE	4.620	6.483	14.541	1.779	1.502	2.509	4.275	0.388	3.995	7.005	10.809	1.070	3.717	5.938	11.043	1.371
JRL	5.378*	7.545*	16.774*	2.085*	1.735*	2.989*	4.634*	0.442*	4.364*	7.510*	10.940*	1.096*	4.396*	6.949*	12.776*	1.546*
PGPR (Ours)	**5.590**	**7.569**	**16.886**	**2.157**	**2.858**	**4.834**	**7.020**	**0.728**	**5.042**	**8.416**	**11.904**	**1.274**	**5.449**	**8.324**	**14.401**	**1.707**

Table 2: Overall recommendation effectiveness of our method compared to other baselines on four Amazon datasets. The results are reported in percentage (%) and are calculated based on the top-10 predictions in the test set. The best results are highlighted in bold and the best baseline results are marked with a star (∗).

multimodal information including images, text and ratings into a neural network. Note that we did not include [28] as a baseline because we are unable to enumerate all the possible paths between user–item pairs due to the large scale of our datasets.

All models are evaluated in terms of four representative top-N recommendation measures: **Normalized Discounted Cumulative Gain (NDCG)**, **Recall**, **Hit Ratio (HR)** and **Precision (Prec.)**. These ranking metrics are computed based on the top-10 predictions for every user in the test set.

Implementation Details. The default parameter settings across all experiments are as follows. For the KGRE-Rec problem, we set the maximum path length to 3 based on the assumption that shorter paths are more reliable for users to interpret the reasons of recommendation. For models' latent representations, the embeddings of all entities and relations are trained based on the 1-hop scoring function defined in Equation 11, and the embedding size is set to 100. On the RL side, the history vector $\mathbf{h_t}$ is represented by the concatenation of embeddings of e_{t-1} and r_t, so the state vector $\mathbf{s_t} = (\mathbf{u}, \mathbf{e_t}, \mathbf{e_{t-1}}, \mathbf{r_t})$ is of size 400. The maximum size of the pruned action space is set to 250, i.e., there are at most 250 actions for any state. To encourage the diversity of paths, we further adopt action dropout on the pruned action space with a rate of 0.5. The discount factor γ is 0.99. For the policy/value network, $\mathbf{W}_1 \in \mathbb{R}^{400\times512}$, $\mathbf{W}_2 \in \mathbb{R}^{512\times256}$, $\mathbf{W}_\text{p} \in \mathbb{R}^{256\times250}$ and $\mathbf{W}_\text{v} \in \mathbb{R}^{256\times1}$. For all four datasets, our model is trained for 50 epochs using Adam optimization. We set a learning rate of 0.001 and a batch size of 64 for the *CDs & Vinyl* dataset, and a learning rate of 0.0001 and batch size of 32 for the other datasets. The weight of the entropy loss is 0.001. In the path reasoning phase, we set the sampling sizes at each step to $K_1 = 20, K_2 = 10, K_3 = 1$ for *CDs & Vinyl*, and $K_1 = 25, K_2 = 5, K_3 = 1$ for the other three datasets.

4.3 Quantitative Analysis

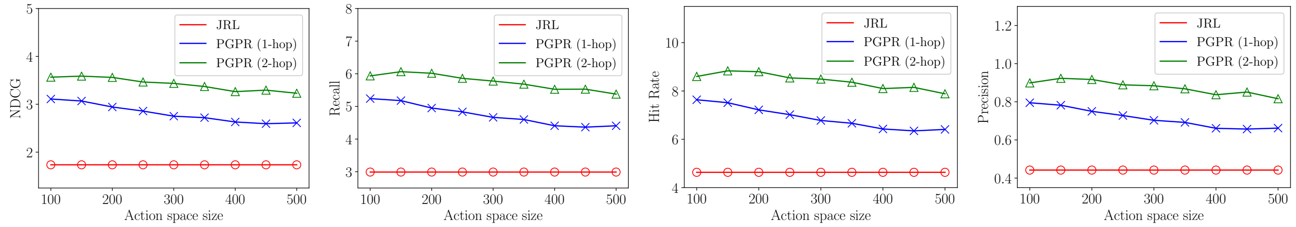
In this experiment, we quantitatively evaluate the performance of our model on the recommendation problem compared to other baselines on all four Amazon datasets. We follow the default setting as described in the previous section.

The results are reported in Table 2. Overall, our *PGPR* method consistently outperforms all other baselines on all datasets in terms of NDCG, Hit Rate, Recall and Precision. For example, it obtains a 3.94% NDCG improvement over the best baseline (JRL) on the *CDs & Vinyl* dataset, a significant improvement of 64.73% on *Clothing*, 15.53% on *Cell Phone*, and 23.95% on *Beauty*. Similar trends can be observed for Recall, Hit Rate and Precision on all datasets. This shows that searching for reasoning paths over the knowledge graph provides substantial benefits for product recommendation and hence—even in the absence of interpretability concerns—is a promising technique for recommendation over graphs. Our path reasoning process is guided by the learned policy network that incorporates rich heterogeneous information from knowledge graphs and captures multi-hop interactions among entities (e.g., user mentions feature, feature is described by item, item belongs to category, etc.). This also largely contributes to the promising recommendation performance of our method. Besides, we notice that directly applying TransE for recommendation [1] slightly outperforms ours. It can be regarded as a single-hop latent matching method, but the post-hoc explanations do not necessarily reflect the true reason of generating a recommendation. In contrast, our methods generate recommendations through an explicit path reasoning process over knowledge graphs, so that the explanations directly reflect how the decisions are generated, which makes the system transparent.

Dataset	# Valid Paths/User	# Items/User	# Paths/Item
CDs & Vinyl	173.38 ± 29.63	120.53 ± 25.04	1.44 ± 1.12
Clothing	60.78 ± 7.00	37.21 ± 7.23	1.63 ± 1.25
Cell Phone	117.22 ± 13.12	57.99 ± 14.29	2.02 ± 2.34
Beauty	59.95 ± 6.28	36.91 ± 7.24	1.62 ± 1.25

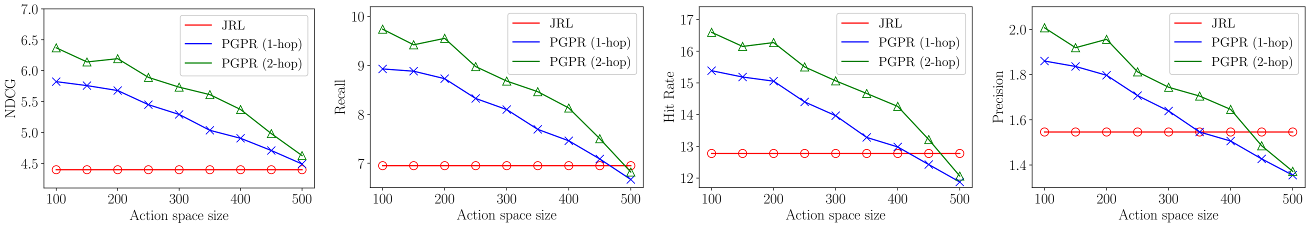
Table 3: Results of number of valid paths per user, number of unique items per user and number of paths per item.

Furthermore, we examine the efficiency of our method in finding valid reasoning paths. A path is deemed *valid* if it starts from a user and ends at an item entity within three hops (i.e., at most four entities in a path). As shown in Table 3, we respectively report the average number of valid paths per user, the average number of unique items per user, and the average number of supportive paths per item. We observe two interesting facts. First, the success rate of our method to find valid paths is around 50%, which is calculated as the number of valid paths out of all sampled paths (200 paths for *CDs & Vinyl* dataset and 125 for others). Especially for the *Cell Phone* dataset, almost all paths are valid. Considering that the number of all possible paths from each user to items is very large and the difficulty of our recommendation problem is particularly high, these results suggest that our method performs very well in regard to path finding properties. Second, each recommended item is associated with around 1.6 reasoning paths. This implies that there are multiple reasoning paths that can serve as supportive evidence for each



(a) NDCG (b) Recall (c) Hit Rate (d) Precision

Figure 3: Recommendation effectiveness of our model under different sizes of pruned action spaces on the *Clothing* dataset. The results using multi-hop scoring function are also reported.



(a) NDCG (b) Recall (c) Hit Rate (d) Precision

Figure 4: Recommendation effectiveness of our model under different sizes of pruned action spaces on the *Beauty* dataset. The results using multi-hop scoring function are also reported.

recommendation. One could hence consider providing more than one of these if users request further details.

4.4 Influence of Action Pruning Strategy

In this experiment, we evaluate how the performance of our model varies with different sizes of pruned action spaces.

Recall that conditioned on the starting user, the action space is pruned according to the scoring function defined in Equation 9, where actions with larger scores are more likely to be preserved. In other words, larger action spaces contain more actions that are less relevant to the user. This experiment aims to show whether larger action spaces are helpful in exploring more reasoning paths to find potential items. We experiment on two selected datasets, *Beauty* and *Clothing*, and follow the default setting from the previous section, except that the size of the pruned action space is varied from 100 to 500 with a step size of 50. The results on two datasets are plotted in Figures 3 and 4, respectively. The best baseline method JRL is also reported in the figures for comparison. Its performance does not depend on the action space.

There are two interesting observations in the results. First, our model outperforms JRL under most choices of pruned action space sizes. Take the *Clothing* dataset as an example. As shown in Figure 3, for any metric among NDCG, Recall, Hit Rate and Precision, the blue curve of our method is consistently above the red curve of JRL by a large margin for all sizes ranging from 100 to 500. The results further demonstrate the effectiveness of our method compared to other baselines. Second, the performance of our model is slightly influenced by the size of the pruned action space. As shown in both figures, the common trend is that a smaller pruned action space leads to better performance. This means that the scoring function is a good indicator for filtering proper actions conditioned on the starting user. Another possible reason is that larger action spaces require more exploration in RL, but for fair comparison, we set the same parameters such as learning rate and training steps across all different choices of action space, which may lead to suboptimal solutions in cases of larger action spaces.

Dataset	Clothing				Beauty			
Sizes	NDCG	Recall	HR	Prec.	NDCG	Recall	HR	Prec.
25, 5, 1	2.858	4.834	7.020	0.728	5.449	8.324	14.401	1.707
20, 6, 1	2.918	4.943	7.217	0.749	5.555	8.470	14.611	1.749
20, 3, 2	2.538	4.230	6.177	0.636	4.596	6.773	12.130	1.381
15, 8, 1	2.988	5.074	7.352	0.767	5.749	8.882	15.268	1.848
15, 4, 2	2.605	4.348	6.354	0.654	4.829	7.138	12.687	1.458
12, 10, 1	3.051	5.207	7.591	0.791	5.863	9.108	15.599	1.905
12, 5, 2	2.700	4.525	6.575	0.679	4.968	7.365	13.168	1.519
10, 12, 1	**3.081**	**5.271**	**7.673**	**0.797**	**5.926**	**9.166**	**15.667**	**1.920**
10, 6, 2	2.728	4.583	6.733	0.693	5.067	7.554	13.423	1.559

Table 4: Influence of sampling sizes at each level on the recommendation quality. The best results are highlighted in bold and the results under the default setting are underlined. All numbers in the table are given in percentage (%).

4.5 Multi-Hop Scoring Function

Besides action pruning and the reward definition, the scoring function is also used as a part of the objective function in training the knowledge graph representation. By default, we employ a 1-hop scoring function for representation learning. In this experiment, we explore whether multi-hop scoring functions can further improve the recommendation performance of our method.

In particular, the 2-hop scoring function from Equation 9 is: $f(e_0, e_2|\tilde{r}_{2,2}) = \langle \mathbf{e_0} + \mathbf{r_1} + \mathbf{r_2}, \mathbf{e_2} \rangle + b_{e_2}$ for any valid 2-hop pattern $\tilde{r}_{2,2} = \{r_1, r_2\}$ between entities e_0 and e_2. This function is plugged into the objective function in Equation 14 for training entity and relation embeddings. All further settings are adopted from the previous action space experiment. We also plot the results in Figures 3

Dataset	Clothing				Beauty			
History	NDCG	Recall	HR	Prec.	NDCG	Recall	HR	Prec.
0-step	1.972	3.117	4.492	0.462	3.236	4.407	8.026	0.888
1-step	2.858	4.834	7.020	0.728	5.449	8.324	14.401	1.707
2-step	2.786	4.702	6.865	0.710	5.342	8.181	14.168	1.669

Table 5: Results for different history representations of state. All numbers in the table are given in percentage (%).

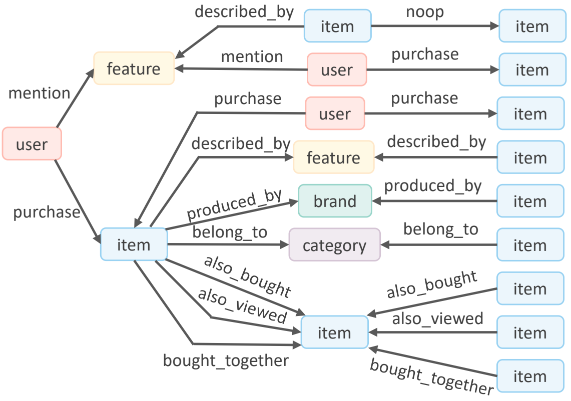


Figure 5: All 3-hop path patterns found in the results.

and 4 with an additional green curve representing our new model trained by the 2-hop scoring function. Surprisingly, we find that our 2-hop PGPR method further outperforms the default model (blue curve). This improvement mainly stems from the effectiveness of the multi-hop scoring function, which captures interactions between entities with longer distance. For example, if a user purchases an item and the item belongs to a category, the 2-hop scoring function enhances the relevance between the *User* entity and the *Category* entity through the 2-hop pattern {*Purchase*, *Belong_to*}.

4.6 Sampling Size in Path Reasoning

In this experiment, we study how the sampling size for path reasoning influences the recommendation performance of our method.

We carefully design 9 different combinations of sampling sizes given a path length of 3. As listed in the first column of Table 4, each tuple (K_1, K_2, K_3) means that the we sample top K_t actions at step t as described in Algorithm 1. For fair comparison, the total number of sampling paths ($= K_1 \times K_2 \times K_3$) is fixed to 120 (except for the first case). We experiment on the *Clothing* and *Beauty* datasets and follow the default settings of other parameters. The recommendation results in terms of NDCG, Recall, Hit Rate and Precision are reported in Table 4. Interestingly, we observe that the first two levels of sampling sizes play a more significant role in finding good paths. For example, in the cases of $(25, 5, 1), (20, 6, 1), (15, 8, 1), (12, 10, 1), (10, 12, 1)$, our model performs much better than in the rest of cases. One explanation is that the first two selections of actions largely determine what kinds of items can be reached. After the first two steps are determined, the policy network tends to converge to selecting the optimal action leading to a good item. On the other hand, our model is quite stable if the sample sizes at the first two levels are large, which offers a good guidance for parameter tuning.

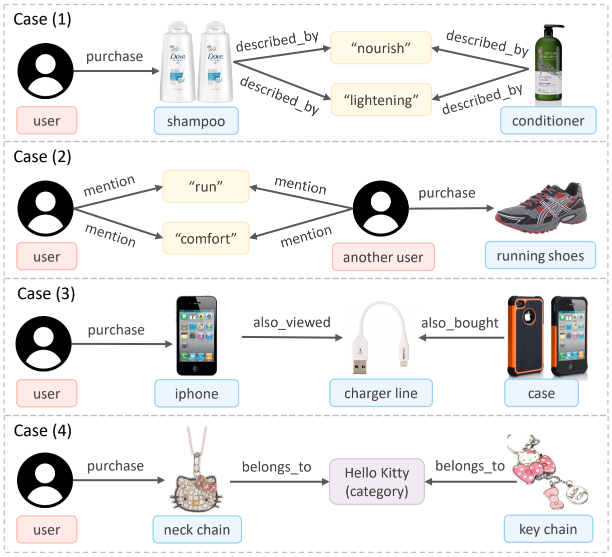


Figure 6: Real cases of recommendation reasoning paths.

4.7 History Representations

Finally, we examine how different representations of state history influence our method. We consider three alternatives for $\mathbf{h}_t$: no history (0-step), last entity e_{t-1} with relation r_t (1-step), and last two entities e_{t-2}, e_{t-1} with relations r_{t-1}, r_t (2-step). Other settings are the same as in the previous experiments. As shown in Table 5, we find that the worst results are obtained in the 0-step case, which suggests that a state representation without history cannot provide sufficient information for the RL agent to learn a good policy. Apart from this, the performance of using 2-step history is slightly worse than that of 1-step history. One possible reason is that additional history information is redundant and even misleads the algorithm in the decision-making process.

5 CASE STUDY ON PATH REASONING

To intuitively understand how our model interprets the recommendation, we give a case study here based on the results generated in the previous experiments. We first study the path patterns discovered by our model during the reasoning process, followed by various cases for recommendation.

Path patterns. For a fixed path length of 3, we find that our method managed to discover 15 different path patterns, which are plotted in an aggregated form in Figure 5. Interestingly, the pattern $\left\{user \xrightarrow{purchase} item \xleftarrow{purchase} user \xrightarrow{purchase} item\right\}$ is one kind of collaborative filtering.

Case study. As shown in Figure 6, we provide several real-world examples of the reasoning paths generated by our method to illustrate how to interpret recommendations through paths.

The first example (Case 1) comes from the *Beauty* dataset, where a user purchased an item "shampoo" that was described by two feature words "nourish" and "lightening". Meanwhile, another item "conditioner" also contained these two features in some review. Therefore, our model recommended "conditioner" to this user. In the second example (Case 2), there are two users who both mentioned the feature words "run" and "comfort" in their reviews, so our method made a decision based on the purchase history of one user, by recommending the item "running shoes" purchased by the user to the other user. In the third example (Case 3), a user bought an item "iPhone" and also viewed another item "charger line". Considering that other users who purchased "phone case" would also buy "charger line", our method accordingly recommended "iPhone case" to the user. The last example (Case 4) depicts that one user purchased an item "neck chain", which belonged to the category of "Hello Kitty". Thus, our method recommended the user another item "key chain" that was also in the same category "Hello Kitty". We conclude that our PGPR method not only achieves promising recommendation results, but also is able to efficiently find diverse reasoning paths for the recommendations.

6 CONCLUSIONS AND FUTURE WORK

We believe that future intelligent agents should have the ability to perform explicit reasoning over knowledge for decision making. In this paper, we propose RL-based reasoning over knowledge graphs for recommendation with interpretation. To achieve this, we develop a method called Policy-Guided Path Reasoning (PGPR). Based on our proposed soft reward strategy, user-conditional action pruning strategy, and a multi-hop scoring approach, our RL-based PGPR algorithm is not only capable of reaching outstanding recommendation results, but also exposes its reasoning procedure for explainability. We conduct extensive experiments to verify the performance of our approach compared with several state-of-art baselines. It should be noted that our PGPR approach is a flexible graph reasoning framework and can be extended to many other graph-based tasks such as product search and social recommendation, which will be explored in the future. We can also extend our PGPR approach to model time-evolving graphs so as to provide dynamic decision support.

REFERENCES

[1] Qingyao Ai, Vahid Azizi, Xu Chen, and Yongfeng Zhang. 2018. Learning Heterogeneous Knowledge Base Embeddings for Explainable Recommendation. *Algorithms* (2018).

[2] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*. 2787–2795.

[3] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2017. Go for a Walk and Arrive at the Answer: Reasoning Over Paths in Knowledge Bases with Reinforcement Learning. In *NIPS-17 Workshop on Automatic Knowledge-Base Construction*.

[4] Li Gao, Hong Yang, Jia Wu, Chuan Zhou, Weixue Lu, and Yue Hu. 2018. Recommendation with multi-source heterogeneous information. *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence* (2018).

[5] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 855–864.

[6] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-based recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 161–169.

[7] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on World Wide Web*. 507–517.

[8] Ruining He and Julian McAuley. 2016. VBPR: Visual Bayesian Personalized Ranking from Implicit Feedback. In *AAAI*.

[9] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *WWW*. 173–182.

[10] Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y Chang. 2018. Improving sequential recommendation with knowledge-enhanced memory networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 505–514.

[11] Joseph A Konstan, Bradley N Miller, David Maltz, Jonathan L Herlocker, Lee R Gordon, and John Riedl. 1997. GroupLens: applying collaborative filtering to Usenet news. *Commun. ACM* 40, 3 (1997), 77–87.

[12] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.

[13] Daniel D Lee and H Sebastian Seung. 2001. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*. 556–562.

[14] Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-Hop Knowledge Graph Reasoning with Reward Shaping. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018, Brussels, Belgium, October 31-November 4, 2018*.

[15] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing* 1 (2003), 76–80.

[16] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 165–172.

[17] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[18] Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic matrix factorization. In *Advances in neural information processing systems*. 1257–1264.

[19] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A Three-Way Model for Collective Learning on Multi-Relational Data.. In *ICML*. 809–816.

[20] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th conference on uncertainty in artificial intelligence*. AUAI Press, 452–461.

[21] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. GroupLens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. ACM, 175–186.

[22] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. ACM, 285–295.

[23] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484.

[24] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[25] Georgios Theocharous, Philip S Thomas, and Mohammad Ghavamzadeh. 2015. Personalized Ad Recommendation Systems for Life-Time Value Optimization with Guarantees.. In *IJCAI*. 1806–1812.

[26] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *CIKM*. ACM, 417–426.

[27] Xiting Wang, Yiru Chen, Jie Yang, Le Wu, Zhengtao Wu, and Xing Xie. 2018. A Reinforcement Learning Framework for Explainable Recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 587–596.

[28] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable Reasoning over Knowledge Graphs for Recommendation. *AAAI* (2019).

[29] Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. *Proceedings of Conference on Empirical Methods in Natural Language Processing* (2017), 564–573.

[30] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative Knowledge Base Embedding for Recommender Systems. In *KDD*.

[31] Wei Zhang, Quan Yuan, Jiawei Han, and Jianyong Wang. 2016. Collaborative Multi-Level Embedding Learning from Reviews for Rating Prediction. In *IJCAI*.

[32] Yongfeng Zhang, Qingyao Ai, Xu Chen, and W Bruce Croft. 2017. Joint representation learning for top-n recommendation with heterogeneous information sources. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 1449–1458.

[33] Yongfeng Zhang and Xu Chen. 2018. Explainable Recommendation: A Survey and New Perspectives. *arXiv preprint arXiv:1804.11192* (2018).

[34] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit Factor Models for Explainable Recommendation based on Phrase-level Sentiment Analysis. *SIGIR* (2014), 83–92.

[35] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. 2018. DRN: A Deep Reinforcement Learning Framework for News Recommendation. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 167–176.

[36] Lei Zheng, Vahid Noroozi, and Philip S. Yu. 2017. Joint deep modeling of users and items using reviews for recommendation. In *WSDM*.