Leveraging Demonstrations for Reinforcement Recommendation Reasoning over Knowledge Graphs

Kangzhi Zhao
Tsinghua University
zkz15@mails.tsinghua.edu.cn

Xiting Wang*
Microsoft Research Asia
xitwan@microsoft.com

Yuren Zhang
University of Science and Technology of China
yr160698@mail.ustc.edu.cn

Li Zhao
Microsoft Research Asia
lizo@microsoft.com

Zheng Liu
Microsoft Research Asia
Zheng.Liu@microsoft.com

Chunxiao Xing
Tsinghua University
xingcx@tsinghua.edu.cn

Xing Xie
Microsoft Research Asia
xing.xie@microsoft.com

ABSTRACT

Knowledge graphs have been widely adopted to improve recommendation accuracy. The multi-hop user-item connections on knowledge graphs also endow reasoning about why an item is recommended. However, reasoning on paths is a complex combinatorial optimization problem. Traditional recommendation methods usually adopt brute-force methods to find feasible paths, which results in issues related to convergence and explainability. In this paper, we address these issues by better supervising the path finding process. The key idea is to extract imperfect path demonstrations with minimum labeling efforts and effectively leverage these demonstrations to guide path finding. In particular, we design a demonstration-based knowledge graph reasoning framework for explainable recommendation. We also propose an ADversarial Actor-Critic (ADAC) model for the demonstration-guided path finding. Experiments on three real-world benchmarks show that our method converges more quickly than the state-of-the-art baseline and achieves better recommendation accuracy and explainability.

CCS CONCEPTS

• **Information systems → Recommender systems**; • **Computing methodologies → Reinforcement learning**.

KEYWORDS

Explainable Recommendation; Reinforcement Learning; Knowledge Graph Reasoning

ACM Reference Format:
Kangzhi Zhao, Xiting Wang, Yuren Zhang, Li Zhao, Zheng Liu, Chunxiao Xing, and Xing Xie. 2020. Leveraging Demonstrations for Reinforcement Recommendation Reasoning over Knowledge Graphs. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401171

*Xiting Wang is the corresponding author.

1 INTRODUCTION

Knowledge graphs (KGs), which organize auxiliary facts about items in heterogeneous graphs, have been shown effective in improving recommendation performance.On the one hand, the connectivity between users and items in a KG helps better model underlying user-item relations and improve recommendation **accuracy**. On the other hand, the multi-hop connections between users and items endow reasoning about recommendations, which enhances **explainability**. For example, the reason for recommending *Acalme Sneaker* to user *Bob* can be revealed by the connection $Bob \xrightarrow{Purchase} Revolution\ 5\ Running\ Shoe \xrightarrow{Produced\_By} Nike \xrightarrow{Produce} Acalme\ Sneaker$.Compared with natural language explanations [11, 14, 24, 52], knowledge graph reasoning seldom makes false statements about items and is able to faithfully reflect the working mechanism of the recommendation model, thus increasing user trust and satisfaction [38].

While knowledge graph reasoning is promising, challenges still exist. Reasoning on paths is a complex combinatorial optimization problem, in which the set of feasible solutions consists of multi-hop paths that connect users with items. Compared with traditional recommendation methods that focus on *scoring a given candidate* according to user preferences (**scoring**), KG reasoning additionally requires *identifying feasible candidate paths* by exploring the knowledge graph (**path finding**). Researchers have designed sophisticated models for scoring candidate paths [35, 43], but the importance of efficient and effective path finding is often overlooked. The time-consuming path finding task is poorly supervised and is usually solved by using brute-force methods, which results in issues with respect to convergence and explainability:

Convergence. It is difficult for existing methods to quickly converge to a satisfying solution because they lack a mechanism to effectively guide and supervise path finding. For example, exhaustive search enumerates all possible candidate paths in the KG [35, 43], which is not applicable to large-scale KGs. REINFORCE with baseline starts with a random path finding policy and gradually improves it by sampling paths to obtain sparse reward signals [46]. This trial-and-error method suffers from poor convergence properties due to the sparsity of the reward signals and the large action space in KGs [47].

Explainability. Since existing methods poorly supervise path finding, there is no guarantee that the discovered paths are highly interpretable.While existing KG reasoning methods optimize only recommendation accuracy, they do not guarantee good interpretability. While all paths that connect a same user-item pair result in the same recommendation accuracy, some paths (reasons) are less convincing. To achieve good explainability, it is important that the user is interested in the entities (e.g., *Nike*) and the relation types (e.g,. *Produce*) involved in the paths. Reasoning with persuasive types of paths is also important for improving explainability. For example, a path like *User A* $\xrightarrow{View}$ *Item A* $\xrightarrow{Viewed\_By}$ *User B* $\xrightarrow{View}$ *Item B* may be less convincing compared with *User A* $\xrightarrow{Purchase}$ *Item A* $\xrightarrow{Purchased\_By}$ *User B* $\xrightarrow{Purchase}$ *Item B*.

The goal of this paper is to study how fast convergence and better explainability can be achieved by better supervising path finding. The major challenge is that we do not have ground-truth paths to enable supervised path finding. While weak supervision can be achieved by using (imperfect) demonstrations of reasoning paths, it is unclear how such demonstrations can be easily obtained. Manually labeling all desirable meta-paths [20] or association rules [7, 27] is a heavy and tedious process, and these meta-paths and rules fail to uncover unseen and personalized connectivity patterns. Even if some demonstrations may be extracted, they are likely to be sparse and noisy, i.e., different from the optimal ground-truth paths. How to leverage these imperfect demonstrations effectively for improving performance remains unsolved.

In this paper, we address the aforementioned issues and show how imperfect path demonstrations can be extracted and leveraged for effective knowledge graph reasoning. To this end, we design a demonstration-based knowledge graph reasoning framework for explainable recommendation. In our framework, a meta-heuristic-based demonstration extractor first derives a set of path demonstrations with minimum labeling efforts. The extraction is guided by desirable properties for demonstrations, which are defined based on meta-heuristics. By effectively leveraging the imperfect demonstrations, an **ADversarial Actor-Critic (ADAC)** path finding model then learns to identify interpretable reasoning paths that lead to accurate recommendations with fast convergence. The ADAC model optimizes the path finding policy by jointly and effectively modeling both demonstrations and reward signals obtained based on historical user preferences. We show how demonstrations can be modeled by using adversarial imitation learning, how the rewards can be accurately estimated by using a critic, and how a tight collaboration between these two parts can be achieved by modeling them in a unified framework.

Our contributions can be summarized as follows:

- We propose to guide knowledge graph reasoning with demonstrations and show how these demonstrations can be extracted with minimum labeling efforts by using our meta-heuristic-based extraction method.
- We propose an Adversarial Actor-Critic model for demonstration-guided path finding, which can identify interpretable reasoning paths that lead to accurate recommendations by effectively leveraging imperfection demonstrations.
- Experiments on real-world benchmarks show that our method converges more quickly than the state-of-the-art baseline, achieves an average recommendation accuracy gain of 6.8%, and increases explainability by 9.3%.

2 PROBLEM FORMULATION

Reasoning is the task of learning explicit inference formulas [47]. In this paper, we consider the formulas as multi-hop paths on the KG. Accordingly, the KG reasoning problem for explainable recommendation can be formulated as follows:

Input. The model input consists of the user set U, the item set V, the observed interactions V_u, and the knowledge graph $\mathcal{G}$:

- Each **user** is denoted by its user ID $u \in U$.
- Each **item** is represented by the item ID $v \in V$.
- The **observed interaction** set V_u for each user u contains all the items the user interacted with in the training set.
- The **knowledge graph** is denoted by $\mathcal{G} = \{(e, r, e') \mid e, e' \in \mathcal{E}, r \in \mathcal{R}\}$, where $\mathcal{E}$ is the entity set and $\mathcal{R}$ is the relation set. Each triplet (e, r, e') indicates that the head entity e (e.g., *Nike*) and the tail entity e' (e.g., *Acalme Sneaker*) are connected by the relation r (e.g., *Produce*). We include observed user-item interactions in the KG to facilitate reasoning by following previous works [43, 46]. Accordingly, we have $U, V \subseteq \mathcal{E}$ and $(u, r^*, v_u) \in \mathcal{G}$, $\forall\, v_u \in V_u$, where $r^* \in \mathcal{R}$ denotes the observed interactions.

Output. Given a user u, our model outputs:

- The **recommended item set** $\widehat{V}_u \subseteq V$.
- A **reasoning path** $\tau_{u,\widehat{v}_u}$ for each recommended item $\widehat{v}_u \in \widehat{V}_u$. $\tau_{u,\widehat{v}_u}$ is a multi-hop path on the KG $\mathcal{G}$ and it connects user u with $\widehat{v}_u$: $\tau_{u,\widehat{v}_u} = [u \xrightarrow{r_1} e_1 \xrightarrow{r_2} \dots \xrightarrow{r_{k-1}} e_{k-1} \xrightarrow{r_k} \widehat{v}_u]$.

3 METHOD

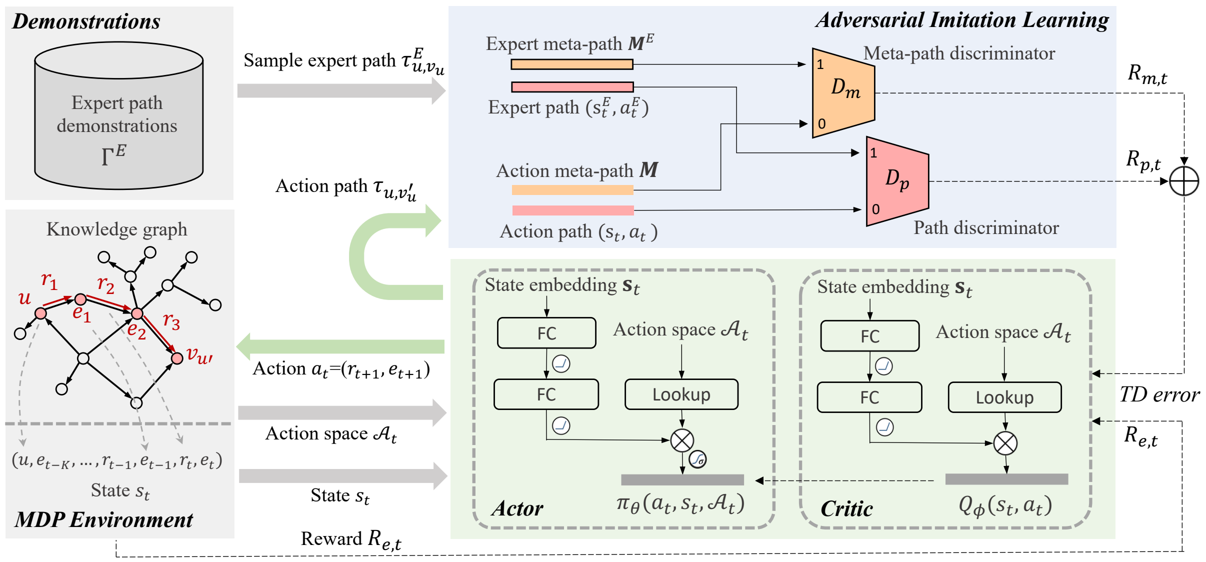
Our demonstration-based knowledge graph reasoning method enables weak supervision by providing imperfect labels for reasoning paths.In particular, our method consists of two parts:a meta-heuristic-based demonstration extraction and a demonstration-guided path finding with the proposed Adversarial Actor-Critic model. The following sections of the paper delve deeper into both parts.

3.1 Demonstration Extraction

Demonstration extraction obtains a set of expert demonstrations $\Gamma^E = \{\tau^E_{u,v_u} | u \in U, v_u \in V_u\}$, where $\tau^E_{u,v_u} = [u \xrightarrow{r^E_1} e^E_1 \xrightarrow{r^E_2} \dots \xrightarrow{r^E_k} v_u]$ is a multi-hop path on the knowledge graph that connects user u with item v_u. To derive imperfect demonstrations that are useful for knowledge graph reasoning, we propose a meta-heuristic-based extraction method.

Meta-heuristics are "*concepts that can be used to define heuristic methods*" and are frequently used to solve combinatorial optimization problems [4]. Compared with methods that enable weak supervision by using expert labels or crowd sourcing [15], meta-heuristics may significantly reduce the number of required manual labels. In this paper, we define meta-heuristics by specifying desirable properties for path demonstrations. These desirable properties are then used to define heuristics for demonstration extraction. In particular, the following three properties are considered.

- *P1: Accessibility.* The demonstrations can be obtained with minimum labeling efforts.
- *P2: Explainability.* The demonstrations are more interpretable than randomly sampled paths.



Demonstrations
Expert path demonstrations Γ^E
Sample expert path τ^E_{u,v_u}
Adversarial Imitation Learning
Expert meta-path $\boldsymbol{M}^E$
Meta-path discriminator
Expert path (s^E_t, a^E_t)
D_m
$R_{m,t}$
Action meta-path $\boldsymbol{M}$
D_p
$R_{p,t}$
Action path (s_t, a_t)
Path discriminator
Action path τ_{u,v'_u}
Knowledge graph
u r_1 e_1 r_2 e_2 r_3 $v_{u'}$
$(u, e_{t-K}, ..., r_{t-1}, e_{t-1}, r_t, e_t)$
State s_t
MDP Environment
Action a_t=(r_{t+1}, e_{t+1})
Action space $\mathcal{A}_t$
State s_t
Reward $R_{e,t}$
State embedding $\mathbf{s}_t$
FC
FC
Action space $\mathcal{A}_t$
Lookup
Actor
$\pi_\theta(a_t, s_t, \mathcal{A}_t)$
Critic
$Q_\phi(s_t, a_t)$
TD error
$R_{e,t}$

Figure 1: Our Adversarial Actor-Critic model for demonstration-guided path finding.

- *P3: Accuracy*. The demonstrations lead to accurate recommendations, i.e., they connect users with items they interacted with.

As long as these three properties are satisfied, the extracted demonstrations are considered useful, even if they are sparse and noisy (not optimal). Following this logic, we define three heuristics for demonstration extraction and explain why each of them satisfies these three properties.

Shortest path. Studies suggest that concise explanations help reduce user cognitive loads and are considered to be more interpretable [18, 23]. Accordingly, we assume that shorter paths between user-item pairs are more explainable than randomly sampled connections (*P2*). To ensure accuracy (*P3*), we only consider paths that connect a user u with items that s/he interacted with (V_u) as demonstrations. In particular, given (u, v_u), we first eliminate the observed interaction between u and v_u from the knowledge graph $\mathcal{G}$. This results in a new KG $\mathcal{G}' = \mathcal{G} \setminus \{(u, r^*, v_u)\}$ where operator "\" performs a set minus. We then regard $\mathcal{G}'$ as an unweighted graph and adopt the Dijkstra's algorithm [12] to automatically generate a shortest path between u and v_u (*P1*), which can be considered as a demonstration. We then repeat this process for $\forall u \in U$ and $\forall v_u \in V_u$ to obtain a set of expert demonstrations.

Meta-path. A meta-path is a sequence of relations between entity types [20]. For example, the meta-path for *Bob* $\xrightarrow{Purchase}$ *Revolution 5 Running Shoe* $\xrightarrow{Produced\_By}$ *Nike* $\xrightarrow{Produce}$ *Acalme Sneaker* can be denoted as *User* $\xrightarrow{Purchase}$ *Item* $\xrightarrow{Produced\_By}$ *Brand* $\xrightarrow{Produce}$ *Item.* In KG reasoning, a meta-path naturally corresponds to a meta-level explanation strategy. Thus, desirable explanation strategies can be indicated by providing meta-paths. Our framework improves the model performance through a very small number (1~3) of manually-defined meta-paths (*P1*). These meta-paths are useful as long as they are considered more interpretable than randomly sampled meta-paths (*P2*). Compared with existing meta-path-based methods [13, 20], our approach requires significantly less labeling effort because it does not require the pre-defined meta-paths to be complete or optimal. We are able to take such imperfect meta-paths as inputs because they are used to *guide* path finding instead of *restricting* the search space. To generate demonstrations based on the pre-defined meta-paths, we simulate constrained random walks [26] on the knowledge graph. In particular, we consider each user u as a starting point of the random walks and sample only the paths whose meta-path belongs to the pre-defined set. Among all sampled paths, only those that lead to items the user interacted with will be kept as expert demonstrations (*P3*).

Path of interest. In addition to adopting a good meta-level explanation strategy, a highly interpretable reasoning path should also fit user interests at the entity level, i.e., it contains entities the users are interested in. In some datasets, obtaining entity-level user interests is relatively easy. For example, in datasets related to user reviews, we can automatically judge (*P1*) if the entities in a path fit user interests by checking whether these entities are mentioned in user reviews. In such a case, we perform random walks to obtain a sampled set of paths, and keep only the paths in which a majority of the entities fit user interests (*P2*). Paths that do not connect a user with items s/he interacted with will be removed to ensure accuracy (*P3*). The remaining paths are considered as expert demonstrations.

The three heuristics result in three different sets of demonstrations. We compare the heuristics empirically by investigating how much their demonstrations help improve the final recommendation accuracy and explainability (Sec. 4.4).

3.2 Adversarial Actor-Critic for Path Finding

Path finding aims to identify a set of recommended items $\widehat{V}_u$ for each user u as well as reasoning paths $\{\tau_{u,\widehat{v}_u} | \widehat{v}_u \in \widehat{V}_u\}$ for the recommendations based on demonstrations Γ^E. A straightforward way to learn a path-finding policy from expert demonstrations is behavior cloning [29], i.e., considering the demonstrations as ground-truth labels and learning a model that tries to generate paths identical to the demonstrations. While this supervised method enables fast convergence, its recommendation accuracy and explainability are limited since the demonstrations are imperfect.

To solve this issue, we propose to effectively leverage both the expert demonstrations Γ^E and knowledge graph $\mathcal{G}$ that contains the observed user interactions $\{(u, r^*, v_u)\}$. The major challenge is how we effectively model the imperfect demonstrations, the observed interactions, and the facts in the KG in a unified framework.To achieve this goal, we design an **ADversarial Actor-Critic model (ADAC)** that integrates actor-critic-based reinforcement learning with adversarial imitation learning.

Fig. 1 shows an overview of our model. The KG is considered a part of the *Markov Decision Process (MDP) environment* [32]. The *actor*, which learns a policy-finding policy, interacts with the MDP environment to obtain its search states on the KG as well as possible actions.The environment informs the actor whether the current policy fits user observed interactions through reward $R_{e,t}$. To integrate the expert demonstrations, an *adversarial imitation learning* component with two discriminators is designed. The discriminators learn to distinguish the expert paths from the paths generated by the actor. The actor tries to "fool" the discriminators by imitating the expert demonstrations. The imitation learning component rewards the actor if its action paths are similar to the expert demonstrations at the meta-path level (high reward $R_{m,t}$) or the path level (high reward $R_{p,t}$). The three types of rewards, i.e., $R_{e,t}$, $R_{m,t}$, and $R_{p,t}$, are jointly modeled by the *critic* to accurately estimate the value of each action. The learned values allow the actor to be trained with an unbiased estimate of the reward gradient. Next, we introduce the major components and describe how they can be jointly optimized in an end-to-end framework.

3.2.1 *MDP Environment.*

We follow the terminologies commonly used in reinforcement learning to describe the MDP environment, which is responsible for informing the actor about its search state in the KG and possible actions to take. The environment also rewards the actor if the current path finding policy fits the observed user interactions. Formally, the MDP environment can be defined by a tuple $(\mathcal{S}, \mathcal{A}, \delta, \rho)$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ is the action space, $\delta : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ refers to the state transition function, and $\rho : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function of the environment.

- **State**: The initial state $s_0 \in \mathcal{S}$ is represented by the user from which we start the path finding process: $s_0 = u$. State $s_t \in \mathcal{S}$ denotes the search status of the actor in the KG at each time t. While embedding the entire KG reasoning history in the state may lead to better model accuracy, we also need to control the model size. To balance accuracy and efficiency, we adopt a partially observed state in our environment, which encodes only the starting user entity and a K-step history of entities and relations, i.e., $s_t = (u, e_{t-K}, ..., r_{t-1}, e_{t-1}, r_t, e_t)$.
- **Action**: For state s_t at each time t, the actor outputs an action $a_t = (r_{t+1}, e_{t+1}) \in \mathcal{A}_t$, where e_{t+1} is the next entity in the path and r_{t+1} is the relation that connects e_t with e_{t+1}. The set of possible actions $\mathcal{A}_t \in \mathcal{A}$ includes all the outgoing neighbors of entity e_t on knowledge graph $\mathcal{G}$ except for the history entities on the path: $\mathcal{A}_t = \{(r, e) \mid r \in \mathcal{R}, e \in \mathcal{E} \setminus \{e_0, e_1, ..., e_{t-1}\}, (e_t, r, e) \in \mathcal{G}\}$.
- **Transition**: We consider the deterministic MDP that the next state s_{t+1} can be deterministically transitioned to as a result of current state s_t and action a_t: $s_{t+1} = \delta(s_t, a_t) = (u, e_{t-K+1}, ..., r_t, e_t, r_{t+1}, e_{t+1})$.
- **Reward**: No intermediate reward is provided during the path finding process. We consider only the terminal reward, which indicates whether the actor generates a path that ends with the items that user u interacted with: $R_{e,T} = \rho(s_T, a_T) = \mathbb{I}_{V_u}(e_T)$. Here, $\mathbb{I}_{V_u}(e_T)$ is an indicator function, i.e., $\mathbb{I}_{V_u}(e_T)$ is 1 when $e_T \in V_u$ and is 0 when $e_T \notin V_u$. This reward measures how much the actor fits the observed user preferences.

3.2.2 *Actor.*

The actor learns a path finding policy π_θ that calculates the probability distribution of the action a_t based on the state s_t and its possible action space $\mathcal{A}_t$: $p(a_t|s_t, \mathcal{A}_t) = \pi_\theta(a_t, s_t, \mathcal{A}_t)$. We model the actor network $\pi_\theta(a_t, s_t, \mathcal{A}_t)$ with fully connected layers and the Softmax function:

$$\boldsymbol{h}_\theta = \mathrm{ReLU}(\boldsymbol{W}_{\theta,1}\mathbf{s}_t) \tag{1}$$

$$p(a_t|s_t, \mathcal{A}_t) = \pi_\theta(a_t, s_t, \mathcal{A}_t) = \frac{\mathbf{a}_t \cdot \mathrm{ReLU}(\boldsymbol{W}_{\theta,2}\boldsymbol{h}_\theta)}{\sum_{a_i \in \mathcal{A}_t} \mathbf{a}_i \cdot \mathrm{ReLU}(\boldsymbol{W}_{\theta,2}\boldsymbol{h}_\theta)} \tag{2}$$

Here, $\mathbf{a}_t \in \mathbb{R}^{d_a}$ is the embedding of the action a_t obtained with a lookup layer, and $\mathbf{s}_t \in \mathbb{R}^{d_s}$ is the embedding of the state s_t. We calculate the state embedding $\mathbf{s}_t$ by concatenating all the entity and relation embeddings of s_t, i.e., $\mathbf{s}_t = \mathbf{u} \oplus \mathbf{e}_{t-K} \oplus ...\mathbf{e}_{t-1} \oplus \mathbf{r}_t \oplus \mathbf{e}_t$, where $\oplus$ denotes the concatenation operator and $\mathbf{u}, \mathbf{r}_t, \mathbf{e}_t \in \mathbb{R}^{d_e}$ are user, relation and entity embeddings learned by using KG embedding techniques [1, 29]. If the length of the path is smaller than K, we pad $\mathbf{s}_t$ with zeros at the end. $\boldsymbol{W}_{\theta,1} \in \mathbb{R}^{d_h \times d_s}$ and $\boldsymbol{W}_{\theta,2} \in \mathbb{R}^{d_a \times d_h}$ are parameters to be learned, and $\mathrm{ReLU}(\cdot)$ is the activation function of the Rectified Linear Unit.

To ensure fast convergence, the actor is initialized by using behavior cloning [29], in which the demonstrations are considered as ground-truth paths to guide the sampling of the actor with Mean Square Error (MSE) loss. The actor is then trained by using our ADAC model, which effectively leverages both the demonstrations and the observed interactions to better guide path finding.

3.2.3 *Adversarial Imitation Learning.*

To jointly model the expert demonstrations with the observed interactions, we leverage Generative Adversarial Imitation Learning [19]. Our adversarial imitation learning component consists of a path discriminator and a meta-path discriminator. The discriminators collaborate with the actor in an adversarial manner: they learn to distinguish the expert paths from the paths generated by the actor, and the actor tries to "fool" the discriminators and obtains good intrinsic rewards by imitating the demonstrations. In particular, discriminators D_p and D_m estimate how likely a path is sampled from the demonstrations at the path level and the meta-path level, respectively.

Path discriminator D_p judges whether the actor can generate a demonstration-like path segment at each time t. The path segment can be represented by s_t and the policy is revealed by a_t. Accordingly, the discriminator is modeled based on s_t and a_t:

$$\boldsymbol{h}_p = \tanh(\mathbf{s}_t \oplus \mathbf{a}_{p,t}) \tag{3}$$

$$D_p(s_t, a_t) = \sigma(\boldsymbol{\beta}_p^T \tanh(\boldsymbol{W}_p \boldsymbol{h}_p)) \tag{4}$$

where $\mathbf{a}_{p,t} \in \mathbb{R}^{d_e}$ is the action embedding of a_t in discriminator D_p. $\tanh(\cdot)$ denotes the hyperbolic tangent function. $\sigma(\cdot)$ is the logistic sigmoid function. $\boldsymbol{W}_p \in \mathbb{R}^{d_a \times (d_s + d_b)}$, $\boldsymbol{\beta}_p \in \mathbb{R}^{d_a}$ are key parameters to be learned.

We train the discriminator so that $D_p(s_t, a_t)$ denotes the probability that (s_t, a_t) comes from a demonstration path. This is achieved by defining the classification loss $\mathcal{L}_\psi$ of D_p as

$$\mathcal{L}_\psi = -(\log D_p(s_t^E, a_t^E) + \log(1 - D_p(s_t, a_t))) \tag{5}$$

In this case, ψ represents the parameters of D_p. $a_t^E = (r_{t+1}^E, e_{t+1}^E)$ and $s_t^E = (u, e_{t-K}^E, ..., r_t^E, e_t^E)$ are determined by the expert path $\tau_{u,v_u}^E = [u \xrightarrow{r_1^E} e_1^E \xrightarrow{r_2^E} ... \xrightarrow{r_k^E} v_u]$.The expert path τ_{u,v_u}^E is randomly sampled from the expert demonstrations that start with u.

The path discriminator rewards the actor if the actor generates (s_t, a_t) pairs that are likely to come from the demonstrations:

$$R_{p,t} = \log(D_p(s_t, a_t)) - \log(1 - D_p(s_t, a_t)) \tag{6}$$

Meta-path discriminator D_m judges whether the overall explanation strategy adopted by the actor is similar to that of the demonstrations by comparing their meta-paths. The meta-path discriminator D_m is similar to the path-discriminator D_p. In many knowledge graphs, a meta-path can be uniquely defined by using its relations[20, 46]. Thus, we define the meta-path embedding $\mathbf{M}$ by concatenating the relation embeddings, i.e., $\mathbf{M} = \mathbf{r}_1 \oplus \mathbf{r}_2 \oplus ... \mathbf{r}_T$. The meta-path discriminator D_m is modeled by

$$\boldsymbol{h}_m = \tanh(\boldsymbol{W}_{m,1}\mathbf{M}) \tag{7}$$

$$D_m(\mathbf{M}) = \sigma(\boldsymbol{\beta}_m^T \tanh(\boldsymbol{W}_{m,2}\boldsymbol{h}_m)) \tag{8}$$

where $\boldsymbol{W}_{m,1} \in \mathbb{R}^{d_h \times (T d_e)}$, $\boldsymbol{W}_{m,2} \in \mathbb{R}^{d_a \times d_h}$, $\boldsymbol{\beta}_m \in \mathbb{R}^{d_a}$ are parameters to be learned.

We train the meta-path discriminator so that $D_m(\mathbf{M})$ denotes the probability that $\mathbf{M}$ corresponds to a meta-path of an expert demonstration. This is achieved by minimizing the following loss:

$$\mathcal{L}_\omega = -(\log D_m(\mathbf{M}^E) + \log(1 - D_m(\mathbf{M}))) \tag{9}$$

where ω represents the paramters of D_m and $\mathbf{M}^E = \mathbf{r}_1^E \oplus \mathbf{r}_2^E \oplus ... \mathbf{r}_T^E$ is the embedding of the expert meta-path.

The reward given by the meta-path discriminator is defined as

$$R_{m,t} = \log(D_m(\mathbf{M})) - \log(1 - D_m(\mathbf{M})) \tag{10}$$

3.2.4 Critic. The critic aims to effectively model the rewards from both reinforcement learning (MDP environment) and imitation learning (discriminators). Since $R_{e,t}$ and $R_{m,t}$ are sparse terminal rewards that can only be obtained after all actions for finding a path have been taken, it is important that we accurately estimate the contribution of each action to the rewards to better guide the actor. To this end, we adopt a critic network [25] for estimating the value (contribution) of each action. Compared with alternatives like using the discounted sum of the rewards to estimate values [45], critic networks have better convergence properties due to lower variance [2]. In particular, our critic network Q_ϕ can calculate the value of each action a_t given state s_t:

$$\boldsymbol{h}_\phi = \mathrm{ReLU}(\boldsymbol{W}_{\phi,1}\mathbf{s}_t) \tag{11}$$

$$Q_\phi(s_t, a_t) = \mathbf{a}_{\phi,t} \cdot \mathrm{ReLU}(\boldsymbol{W}_{\phi,2}\boldsymbol{h}_\phi) \tag{12}$$

where $\boldsymbol{W}_{\phi,1} \in \mathbb{R}^{d_h \times d_s}$ and $\boldsymbol{W}_{\phi,2} \in \mathbb{R}^{d_a \times d_h}$ are the parameters to be learned. $\mathbf{a}_{\phi,t} \in \mathbb{R}^{d_a}$ is the embedding of the action a_t in the critic.

We adopt the Temporal Difference (TD) method [36] to learn the critic network. This method first calculates the target q_t according to the Bellman equation [3]

$$q_t = R_t + \mathbb{E}_{a \sim \pi_\theta} Q_\phi(s_{t+1}, a) \tag{13}$$

where R_t is an aggregated reward that motivates the policy to find paths similar to the demonstrations as well as achieves better recommendation accuracy:

$$R_t = \alpha_p R_{p,t} + \alpha_m R_{m,t} + (1 - \alpha_p - \alpha_m) R_{e,t} \tag{14}$$

where $\alpha_p \in [0, 1]$ and $\alpha_m \in [0, 1 - \alpha_p]$ are the weights of the path discriminator and meta-path discriminator rewards respectively. Then the critic is updated by minimizing the TD error

$$\mathcal{L}_\phi = (Q_\phi(s_t, a_t) - q_t)^2 \tag{15}$$

Given $Q_\phi(s_t, a_t)$, the actor can be learned by minimizing the following loss function:

$$\mathcal{L}_\theta = -\mathbb{E}_{a \sim \pi_\theta} Q_\phi(s_t, a) \tag{16}$$

3.2.5 Joint Learning. We can jointly optimize the actor π_θ, critic Q_ϕ, discriminators D_p and D_m by minimizing combined loss:

$$\mathcal{L} = \mathcal{L}_\theta + \mathcal{L}_\phi + \mathcal{L}_\psi + \mathcal{L}_\omega \tag{17}$$

Since we have $\partial\mathcal{L}/\partial\eta = \partial\mathcal{L}_\eta/\partial\eta$ for $\forall\eta \in \{\theta, \phi, \psi, \omega\}$, we can optimize the joint loss by minimizing $\mathcal{L}_\theta$, $\mathcal{L}_\phi$, $\mathcal{L}_\psi$, $\mathcal{L}_\omega$ successively in one iteration. More specifically, during the m-th iteration, the m-th user is selected as the starting point (initial state s_0) for path finding. We can then sample an action path τ_{u,v'_u} from the actor as well as an expert demonstration τ_{u,v_u}^E from Γ^E. Next, $\mathcal{L}_\theta$, $\mathcal{L}_\phi$, $\mathcal{L}_\psi$, $\mathcal{L}_\omega$ are minimized successively based on τ_{u,v'_u} and τ_{u,v_u}^E. After $|U|$ iterations, actions about all users are tested and one training epoch is completed. We then go to the next training epoch until the model converges or the maximum number of epochs is reached.

4 EXPERIMENT

In this section, we show that our method consistently outperforms the state-of-the-art baselines in terms of both recommendation accuracy and explainability (Sec. 4.2). Ablation study and parameter sensitivity analysis are also conducted to demonstrate the effectiveness of our major model components, including the critic and the discriminators. We also investigate the effects of using different demonstrations and present a case study of reasoning paths generated with our method (Sec. 4.4).

4.1 Experiment Setup

4.1.1 Dataset. We evaluate our model with three datasets: **Beauty**, **Clothing** and **Cell_Phones**. They are from three product categories of *Beauty*, *Clothing Shoes and Jewelery* and *Cell Phones and Accessories* from Amazon 5-core[1]. Each dataset contains reviews, product metadata and links. The review data consists of user-item ratings and corresponding review texts. The product metadata includes the categories and brands. The link data includes the also viewed/also bought graphs. Following previous studies [1, 46, 51],

[1]http://jmcauley.ucsd.edu/data/amazon

we construct the knowledge graph based on the above auxiliary facts and randomly sample 70% of the interactions of each user as the training set and the remaining 30% as the test set. The statistics of the datasets are summarized in Table 1.

Dataset	Beauty	Clothing	Cell_Phones
#Users	22,363	39,387	27,879
#Items	12,101	23,033	10,429
#Interactions	198,502	278,677	194,439
#Relation Types	8	8	8
#Entity Types	6	6	6
#Entities	224,074	425,528	163,249
#Triplets	7,832,720	10,671,090	6,299,494

Table 1: The statistics of our datasets.

4.1.2 Evaluation Metrics. To evaluate the recommendation performance of ADAC, we adopt some widely-used metrics, including Precision (**Precision**), Recall (**Recall**), Normalized Discounted Cumulation Gain (**NDCG**) and Hit Ratio (**HR**). Higher scores in the above metrics indicate better recommendation performance. For each user, we regard all the possible items as candidate items instead of using negative sampling. We evaluate the metrics based on the top-10 recommended items for each user in the test set.

To evaluate the explainability of the reasoning paths, we design two criteria by leveraging the ground-truth reviews. The basic idea is that the ground-truth reviews reveal the reason for the user-item interaction. Thus, if a reasoning path contains many entities that are mentioned in the ground-truth review, then it will achieve good explainability. Specifically, for each positive recommended item $\widehat{v}_u$, we filter a review word if its frequency is more than 5000 or its TF-IDF score is less than 0.1, and the remaining words are considered as ground-truth words. Then we aggregate the entities in the path $\tau_{u,\widehat{v}_u}$ and rank them based on their frequencies. We evaluate the explainability by matching the entities in the reasoning paths with the ground truth words. Entities whose types are *Word*, *Brand*, or *Category* are all mapped to ground-truth words by using string matching. We evaluate the explainability with the metrics **Precision** and **Recall** based on the top-5 matched entities.

4.1.3 Comparison Methods. We compare our ADAC with the following state-of-the-art recommendation methods.

- **BPR** [30]: Bayesian Personalized Ranking (BPR) is a pairwise ranking algorithm to learn the latent representations of users and items for top-N recommendation.
- **RippleNet** [39]: RippleNet is a KG-based recommendation model which incorporates deep neural network to propagate users' potential preferences on the knowledge graph.
- **DKN** [40]: Deep Knowledge-aware Network (DKN) is a state-of-the-art KG-based news recommendation framework to fuse semantic-level and knowledge-level representation. We apply the knowledge-level part in our product recommendation problem.
- **RuleRec** [27]: RuleRec is another state-of-the-art algorithm that incorporates knowledge graphs in recommendation. It constructs a rule-guided model to recommend items based on the induced rules.
- **PGPR** [46]: Policy-Guided Path Reasoning (PGPR) is the state-of-the-art path reasoning algorithm of the knowledge graph-based recommendation. It adopts a policy-based method to search and recommend items on the knowledge graph.
- **ActorCritic** [25]: Actor-Critic takes advantage of both value-based and policy-based methods. We implement it by following the approach in [25] and adopt the behavior cloning initialization used in our model to guide the path reasoning on the knowledge graph.
- **ADAC-C**: ADAC-C removes the critic from ADAC. The aggregated reward is learned through the policy-gradient method as in the PGPR approach.
- **ADAC-P**: ADAC-P removes the path discriminator from ADAC.
- **ADAC-M**: ADAC-P removes the meta-path discriminator.

In order to do a fair comparison with the baselines, we adopt the shortest paths as expert demonstrations. However, we also show that adopting dataset-specific heuristic paths further boosts the performance of our model. We compare different demonstration paths in terms of accuracy and explainability in Sec. 4.4.

4.1.4 Implementation and Training. We mainly refer to Xian et al. [46] to implement our MDP environment. We set the history length $K = 1$ for the state s_t and the maximum length $T = 3$ for the reasoning path τ. We prune the action space with the maximum size 250 and train the KG with the embedding size $d_e = 100$. After we train the policy π_θ, we adopt the probabilistic beam search to reason paths on knowledge graphs. We add a special relation $\varnothing$ which represents the "no operation" action and allows the actor to stay at an entity at any time when searching forward on the knowledge graphs. We set the dimension of action embeddings $d_a = 256$ and the weight matrices of neural networks $d_h = 512$ and $d_s = 400$. We set the rewards weights $\alpha_p = 0.006$ for Beauty, $\alpha_p = 0.004$ for Clothing and Cell_Phones, $\alpha_m = 0.01$ for all the datasets. We also assign $R_{e,T} = 0$ for the path which contains only the user and the items s/he interacted with to avoid overfitting. In the training procedure, the action dropout rate is 0.5. We initialize all the parameters of our neural networks with Xavier initialization [17] and leverage the Adam optimization [22] with the learning rate of 0.0001.

4.2 Overall Performance

Recommendation accuracy. We evaluate the recommendation performance of our model in Table 2. The experimental results show that ADAC outperforms the baselines on all the three datasets. For example, on Beauty, ADAC achieves a precision rate of 1.991% while BPR reaches 1.066%, which means our path reasoning algorithm ADAC can effectively integrate the auxiliary information from the knowledge graph. Compared with the 1.79% precision rate achieved by the most competitive baseline, ADAC has proved to do path reasoning in a more efficient way. Previous models like PGPR and ActorCritic suffer from an inefficient search strategy. The sparse reward signals generate too much noise in their reasoning paths, which makes them hard to exploit for the informative multi-hop relations on the knowledge graph. ADAC uses the shortest paths as demonstrations to search paths towards positive items, which makes it easier for the policy to exploit the historical user preferences. Note that we consider all the items as candidates by following the previous study [46]. Thus, the accuracy scores are smaller compared with evaluations that use negative sampling.

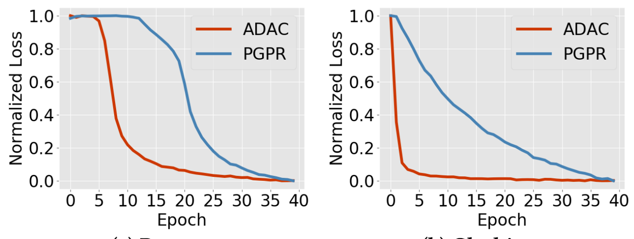
Explainability. Table 3 compares the explainability of different path reasoning methods in terms of precision and recall. We focus

Dataset	Beauty				Clothing				Cell_Phones			
Metrics	**Precision**	**Recall**	**NDCG**	**HR**	**Precision**	**Recall**	**NDCG**	**HR**	**Precision**	**Recall**	**NDCG**	**HR**
BPR	1.066	4.927	2.704	9.113	0.196	1.086	0.598	1.801	0.624	3.363	1.892	5.323
Ripple Net	1.133	5.251	2.458	9.224	0.201	1.112	0.627	1.885	0.688	3.858	1.935	5.727
DKN	1.03	2.489	1.851	8.6	0.106	0.727	0.279	1.012	0.465	3.187	1.603	4.484
RuleRec	1.152	5.213	2.872	9.751	0.21	1.15	0.639	1.912	0.674	3.565	1.966	5.669
PGPR	1.736	8.448	5.511	14.642	0.723	4.827	2.871	7.023	1.274	8.416	5.042	11.904
ActorCritic	1.79	8.764	5.734	15.003	0.72	4.776	2.803	6.965	1.24	8.218	4.888	11.544
ADAC-C	1.924	9.294	6.034	15.522	0.759	4.992	2.957	7.296	1.331	8.836	5.194	12.205
ADAC-P	1.901	9.076	5.854	15.406	0.754	4.971	2.956	7.273	1.295	8.672	5.112	12.107
ADAC-M	1.824	8.809	5.708	14.932	0.745	4.916	2.885	7.214	1.308	8.691	5.122	12.078
ADAC	**1.991**	**9.424**	**6.08**	**16.036**	**0.783**	**5.152**	**3.048**	**7.502**	**1.358**	**8.943**	**5.22**	**12.537**
Imp. (%)	+11.2	+7.5	+6.0	+6.9	+8.3	+6.7	+6.2	+6.8	+6.6	+6.3	+3.5	+5.3

Table 2: Comparison of recommendation accuracy on three real-word datasets. The results are reported in percentage.

Dataset	Beauty		Clothing		Cell_Phones	
Metrics	**Prec.**	**Rec.**	**Prec.**	**Rec.**	**Prec.**	**Rec.**
PGPR	8.507	2.843	9.915	3.108	8.301	2.902
ActorCritic	7.671	2.584	9.722	3.25	8.691	2.82
ADAC-C	8.617	2.639	9.895	3.144	8.82	2.903
ADAC-P	8.774	2.887	10.317	3.356	10.037	2.991
ADAC-M	**9.46**	2.995	10.659	3.317	9.895	3.02
ADAC	9.447	**2.999**	**10.667**	**3.359**	**10.314**	**3.181**
Imp. (%)	+11.0	+5.5	+7.6	+3.4	+18.7	+9.6

Table 3: Comparison of explainability on three real-word datasets. The results are reported in percentage.



(a) Beauty (b) Clothing

Figure 2: Comparison of convergence. Both losses are normalized to $[0, 1]$ for better readability.

only on PGPR and ActorCritic because other baselines cannot generate reasoning paths. The results show that ADAC outperforms the baselines in all three datasets. For example, ADAC achieves a precision rate of 10.314% in the Cell_Phone dataset while ActorCritic stands at 8.691%. This means that leveraging the expert demonstrations can improve explainability even if shortest paths are used as demonstrations. Sec. 4.4 shows the performance of different demonstration paths in terms of explainability.

Convergence. In Fig. 2, we also compare ADAC and the state-of-the-art path reasoning model PGPR for convergence by using the data from Beauty and Clothing. We show only the result of $\mathcal{L}_\theta$ for ADAC because the actor is the policy we use for the path reasoning and the convergence pattern of other losses is similar. We normalize all the losses to $[0, 1]$ for better readability. The result shows that ADAC can efficiently converge with less than 20 epochs with the help of demonstrations. The reason is that our design can effectively integrate the different aspects of the demonstrations.

4.3 Importance of Different Components

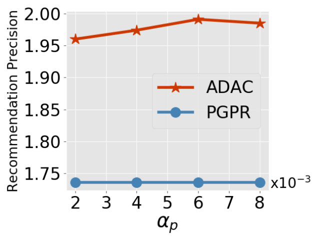
4.3.1 Effectiveness of the Critic. Table 2 and 3 compare ADAC-C and ADAC in terms of recommendation performance and explainability. ADAC-C removes the critic from ADAC and learns to update the model by using only the policy gradient algorithm, which is the same one used in PGPR. ADAC consistently outperforms ADAC-C because the critic can effectively model the rewards from both the reinforcement learning and imitation learning. This demonstrates the effectiveness of the critic network.

4.3.2 Effectiveness of Path Discriminator. To evaluate the effectiveness of the path discriminator D_p, we compare ADAC with ADAC-P, a variant of our method that does not use the path discriminator. The path discriminator learns the probability that a sampled path comes from demonstrations. As shown in Table 2 and 3, ADAC-P performs consistently worse than ADAC, which means the path discriminator can improve both recommendation accuracy and explainability.

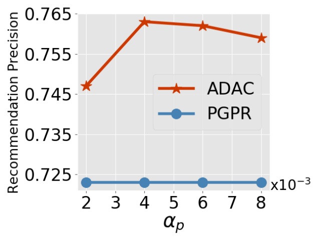
4.3.3 Effectiveness of Meta-path Discriminator. ADAC-M removes the meta-path discriminator from ADAC. The meta-path discriminator D_M introduces $R_{m,t}$ to R_t in order to judge whether the overall strategy of the actor is similar to the demonstration. The comparison between ADAC-M and ADAC shows that meta-path discriminator is effective in achieving high accuracy and explainability, especially in the recommendation task.

4.3.4 Influence of Discriminator Weight. The weights of the path discriminator and meta-path discriminator are α_p and α_m, respectively. Fig. 3 shows the parameter sensitivity of α_p and α_m with regard to the datasets Beauty and Clothing. We provide only the results related to the metric Precision because all the other metrics change according to its value.

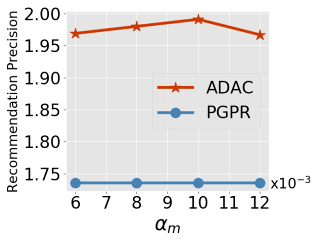
The results lead to two conclusions. First, ADAC performs better than PGPR in both datasets when α_p ranges from 0.002 to 0.008 and α_m ranges from 0.006 to 0.012. Second, the performance of ADAC is slightly influenced by the discriminator weights. The reason is that different weights mean different strategies to exploit the demonstrations. As a result, the performance varies with the



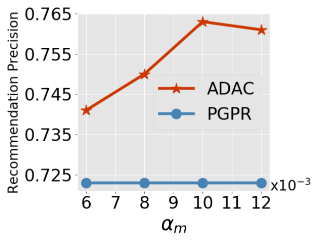
(a) α_p on Beauty



(b) α_p on Clothing



(c) α_m on Beauty



(d) α_m on Clothing

Figure 3: Sensitivity analysis of α_p and α_m on Beauty and Clothing

	Explanation		Recommendation	
	Precision	**Recall**	**Precision**	**Recall**
PGPR	8.507	2.843	1.736	8.448
ActorCritic	7.671	2.584	1.79	8.764
SP	9.447	2.999	1.991	9.424
PI	11.873	3.478	1.94	9.314
U-I-U-I	12.895	3.657	2.13	9.661
U-W-U-I	11.339	3.295	1.346	5.662
U-I-W-I	8.303	2.779	1.763	8.559

Table 4: Explanation and recommendation performance on Beauty with different demonstration paths.

	Explanation		Recommendation	
	Precision	**Recall**	**Precision**	**Recall**
PGPR	9.915	3.108	0.723	4.827
ActorCritic	9.722	3.25	0.72	4.776
SP	10.667	3.359	0.763	5.027
PI	11.97	3.475	0.766	5.08
U-I-U-I	10.57	4.131	0.876	5.596
U-W-U-I	15.185	5.005	0.447	2.932
U-I-W-I	10.841	4.266	0.691	4.51

Table 5: Explanation and recommendation performance on Clothing with different demonstration paths.

datasets. The experimental results also show that ADAC achieves the best recommendation performance when $\alpha_p = 0.006$ in Beauty and $\alpha_p = 0.004$ in Clothing. In both datasets, the performance of ADAC is optimal when $\alpha_m = 0.01$.

4.4 Effects of Using Different Demonstrations

Table 4 and 5 show the accuracy and explainability of ADAC on Beauty and Clothing using different types of demonstrations. Shortest Path (**SP**) is the method used in Sec. 4.2 to compare its results with those of the baselines.Path of Interest (**PI**) uses the *Word* entity from the user's comments to identify entities that the user likes and then includes these entities in the demonstrations. We extract the demonstrations by using three meta-paths. **U-I-U-I** denotes $User \xrightarrow{Purchase} Item \xrightarrow{Purchased\_By} User \xrightarrow{Purchase} Item$. **U-W-U-I** is $User \xrightarrow{Mention} Word \xrightarrow{Mentioned\_By} User \xrightarrow{Purchase} Item$. **U-I-W-I** is $User \xrightarrow{Purchase} Item \xrightarrow{Described\_By} Word \xrightarrow{Describe} Item$.

Results of PI. The results suggest that ADAC can achieve a recommendation performance similar to that of SP by using the paths of interest as demonstrations. At the same time, PI achieves much higher explainability results compared with SP. This shows that ADAC can effectively leverage the entity-level user interest information in the PI demonstrations to improve explainability.

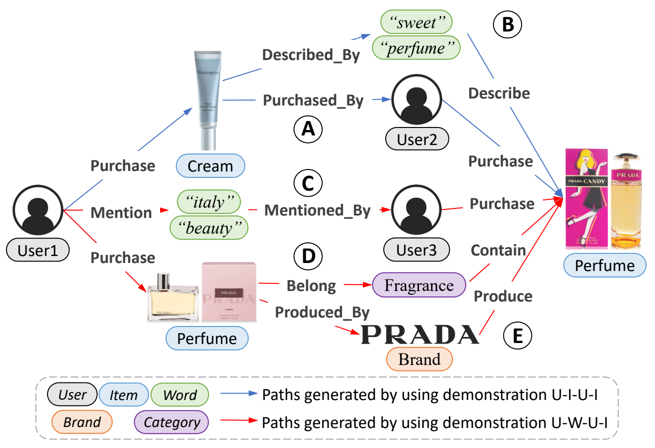


Figure 4: Reasoning paths generated by using different demonstrations.

Results of U-I-U-I. It is interesting that simple meta-paths like U-I-U-I can perform better than SP in terms of both recommendation accuracy (on average +8.91%) and explainability (on average +20.13%). The improvement of recommendation accuracy is expected, considering the success of collaborative filtering methods [33]. However, it is initially difficult to understand why U-I-U-I can also improve explainability. By checking the results, we find that our method can generalize U-I-U-I to other similar but more interpretable meta-paths, e.g., U-I-W-I. Fig. 4 shows examples of reasoning paths generated by using demonstration U-I-U-I (blue). While ADAC is able to find the reasoning path of the type U-I-U-I (Fig. 4A), it can also generalize from U-I-U-I to U-I-W-I (Fig. 4B), which well explains why the user likes the two related items (because of "sweet" and "perfume"). This indicates that by correctly modeling users' item-level interest at the beginning of the training process using U-I-U-I, our model can gradually learn to model users' feature-level interest (U-I-W-I) effectively based on the connections on the knowledge graph. This generalization capability is achieved by using discriminators to guide path finding. In comparison, existing methods that use meta-paths to restrict the search space [20, 53] can hardly achieve this. U-I-U-I can sometimes achieve even better explainability than PI, which may be caused by the sparsity of the PI demonstrations.

Results of U-W-U-I and U-I-W-I. Except for U-I-U-I, other meta-paths such as U-W-U-I can also be generalized by using our model. Fig. 4 shows reasoning paths generated by using U-W-U-I (red). Except for the reasoning path of type U-W-U-I (Fig. 4C), other interpretable paths of type U-I-C-I (Fig. 4D) and U-I-B-I (Fig. 4E) can also be found. Here, C and B represent entity types *Category* and *Brand*, respectively. While U-W-U-I has relatively good explainability, it results in bad recommendation accuracy (Tables 4 and 5), and its performance varies across datasets. Similarly, the performance of U-I-W-I also varies across datasets. These results demonstrate the difficulty of using word-based meta-paths. We suspect that this difficulty may be caused by the ambiguity of the words and the randomness caused by the sample-based extraction method. While a purchased item provides explicit information on user interests, a word mentioned in reviews can be understood in many different ways. Thus, randomly sampling an individual word in the reviews may introduce much noise into the demonstrations.

5 RELATED WORK

Our work is related to the knowledge-graph-based (KG-based) recommendation and reinforcement learning (RL) in recommendation.

5.1 KG-Based Recommendation

Existing knowledge-graph-based recommendation methods can be divided into two groups: embedding-based and path-based.

Embedding-based methods learn entity and relation representations with KG embedding techniques [5, 44] and integrate the learned representations into the recommendation model to improve accuracy [10, 28, 49]. For example, Huang et al. [21] captured the attribute-level user preferences and improved recommendation accuracy by using memory networks to incorporate KG representations. Wang et al. [40] fused the knowledge-level and semantic-level representations of news items to improve the prediction of their click-through rates. Cao et al. [6] devised a system that transferred knowledge by jointly learning a recommendation model and a KG completion model. These methods demonstrate the usefulness of KG in improving recommendation accuracy and illustrate how KG embeddings allow for the flexible incorporation of knowledge. However, their indirect utilization of the knowledge graph structure prevents them from effectively modeling the connectivity patterns and sequential dependencies. As a result, their recommendation accuracy and reasoning capability are limited. For example, they cannot explain their recommendation by providing a path in the KG that connects a user with the recommended item.

Path-based methods learn to recommend by explicitly modeling the sequential connectivity patterns (paths) between items or user-item pairs. Researchers model paths by using methods such as probabilistic logic systems [7], recurrent neural networks [35, 43], and memory networks [39]. For example, Wang et al. [43] learned user preferences by modeling all qualified paths between users and items based on recurrent neural networks. Wang et al. [39] propagated user preferences on knowledge graphs by using memory networks. Ma et al. [27] computed the probability of finding paths based on random walks. Most existing path-based methods focus on modeling *given candidate paths* and leverage brute-force methods for *finding* candidate paths on knowledge graphs, i.e., enumerating all the paths on KGs. As a result, these methods cannot be scaled up to create large knowledge graphs. Recently, a Policy-Guided Path Reasoning algorithm [46] has been proposed, which guides the path-finding policy with sparse reward signals. While this algorithm is more efficient than brute-force searches, the huge action space and the sparse reward make it difficult to quickly converge to a satisfying solution [47]. This issue could be alleviated by pruning the search space using pre-defined desirable connectivity patterns such as meta-paths [13, 20, 34, 48], meta-graphs [53], and association rules [7, 27]. However, these meta-level structures are usually difficult to obtain (e.g., they require heavy and tedious manual labeling) and fail to uncover unseen and personalized connectivity patterns. We aim to solve the aforementioned issues by proposing a demonstration-based knowledge graph reasoning framework. We show how path demonstrations can be extracted with minimum labeling efforts and how imperfect demonstrations can be utilized to ensure fast convergence and improve explainability.

5.2 RL in Recommendation

Reinforcement learning has been adopted to deal with various recommendation tasks, such as news recommendation [56], page-wise recommendation [54] and explainable recommendation [42]. Modeling with MDP allows recommender systems to consider current and future rewards simultaneously and model dynamic user preference [8, 41, 55]. Recently, many advanced reinforcement learning models are adopted by recommender systems.Examples include multi-agent reinforcement learning [16], the hierarchical reinforcement learning [50], and the Generative Adversarial Networks [9, 31].

Most existing RL-based recommender systems are not knowledge-aware. The model most similar to ours is the Policy-Guided Path Reasoning [46], which enables knowledge graph reasoning by incorporating REINFORCE with the baseline [37]. Compared with previous KG-based recommendation models, this model improves recommendation accuracy and is more efficient. However, its path finding process relies on a sparse reward signal and is poorly supervised, which results in issues related to convergence and explainability. Our model addresses these issues by proposing a demonstration-based knowledge graph framework and an Adversarial Actor-Critic model for path finding.

6 CONCLUSION

In this paper, we design a demonstration-based knowledge graph reasoning framework for explainable recommendation, which addresses the issues related to convergence and explainability. We first leverage a meta-heuristic-based demonstration extractor to derive a set of path demonstrations with minimum labeling efforts. Then we propose an ADversarial Actor-Critic (ADAC) model for demonstration-guided path finding. Experiments show that our method outperforms the state-of-the-art baselines on both recommendation accuracy and explainability. In the future, we will investigate how to leverage the reasoning paths to generate natural language explanations for the users.

ACKNOWLEDGMENTS

This work was supported by NSFC (91646202), National Key R&D Program of China (2018YFB1404401, 2018YFB1402701).

REFERENCES

[1] Qingyao Ai, Vahid Azizi, Xu Chen, and Yongfeng Zhang. 2018. Learning Heterogeneous Knowledge Base Embeddings for Explainable Recommendation. *Algorithms* 11, 9 (2018), 137.

[2] Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017. An Actor-Critic Algorithm for Sequence Prediction. In *ICLR (Poster)*.

[3] R Bellman. 2013. Dynamic Programming, Courier Corporation. *New York, NY* 707 (2013).

[4] Christian Blum and Andrea Roli. 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM computing surveys (CSUR)* 35, 3 (2003), 268–308.

[5] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *NIPS*. 2787–2795.

[6] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019. Unifying Knowledge Graph Learning and Recommendation: Towards a Better Understanding of User Preferences. In *WWW*. ACM, 151–161.

[7] Rose Catherine and William W. Cohen. 2016. Personalized Recommendations using Knowledge Graphs: A Probabilistic Logic Programming Approach. In *RecSys*. ACM, 325–332.

[8] Shi-Yong Chen, Yang Yu, Qing Da, Jun Tan, Hai-Kuan Huang, and Hai-Hong Tang. 2018. Stabilizing Reinforcement Learning in Dynamic Environment with Application to Online Recommendation. In *KDD*. ACM, 1187–1196.

[9] Xinshi Chen, Shuang Li, Hui Li, Shaohua Jiang, Yuan Qi, and Le Song. 2019. Generative Adversarial User Model for Reinforcement Learning Based Recommendation System. In *ICML*. PMLR, 1052–1061.

[10] Zhongxia Chen, Xiting Wang, Xing Xie, Mehul Parsana, Akshay Soni, Xiang Ao, and Enhong Chen. 2020. Towards Explainable Conversational Recommendation. In *IJCAI*.

[11] Zhongxia Chen, Xiting Wang, Xing Xie, Tong Wu, Guoqing Bu, Yining Wang, and Enhong Chen. 2019. Co-attentive multi-task learning for explainable recommendation. In *IJCAI*. 2137–2143.

[12] Edsger W Dijkstra et al. 1959. A note on two problems in connexion with graphs. *Numerische mathematik* 1, 1 (1959), 269–271.

[13] Shaohua Fan, Junxiong Zhu, Xiaotian Han, Chuan Shi, Linmei Hu, Biyu Ma, and Yongliang Li. 2019. Metapath-guided Heterogeneous Graph Neural Network for Intent Recommendation. In *KDD*. ACM, 2478–2486.

[14] Jingyue Gao, Xiting Wang, Yasha Wang, and Xing Xie. 2019. Explainable Recommendation Through Attentive Multi-View Learning. AAAI.

[15] Thomas R Gruber, Adam J Cheyer, and Donald W Pitschel. 2016. Crowd sourcing information to fulfill user requests. US Patent 9,280,610.

[16] Tao Gui, Peng Liu, Qi Zhang, Liang Zhu, Minlong Peng, Yunhua Zhou, and Xuanjing Huang. 2019. Mention Recommendation in Twitter with Cooperative Multi-Agent Reinforcement Learning. In *SIGIR*. ACM, 535–544.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *ICCV*. IEEE Computer Society, 1026–1034.

[18] Jonathan L Herlocker, Joseph A Konstan, and John Riedl. 2000. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*. ACM, 241–250.

[19] Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. In *NIPS*. 4565–4573.

[20] Binbin Hu, Chuan Shi, Wayne Xin Zhao, and Philip S. Yu. 2018. Leveraging Meta-path based Context for Top- N Recommendation with A Neural Co-Attention Model. In *KDD*. ACM, 1531–1540.

[21] Jin Huang, Wayne Xin Zhao, Hong-Jian Dou, Ji-Rong Wen, and Edward Y. Chang. 2018. Improving Sequential Recommendation with Knowledge-Enhanced Memory Networks. In *SIGIR*. ACM, 505–514.

[22] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR (Poster)*.

[23] Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. *arXiv preprint arXiv:1606.04155* (2016).

[24] Piji Li, Zihao Wang, Zhaochun Ren, Lidong Bing, and Wai Lam. 2017. Neural rating regression with abstractive tips generation for recommendation. In *SIGIR*. 345–354.

[25] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2016. Continuous control with deep reinforcement learning. In *ICLR (Poster)*.

[26] László Lovász et al. 1993. Random walks on graphs: A survey. *Combinatorics, Paul erdos is eighty* 2, 1 (1993), 1–46.

[27] Weizhi Ma, Min Zhang, Yue Cao, Woojeong Jin, Chenyang Wang, Yiqun Liu, Shaoping Ma, and Xiang Ren. 2019. Jointly Learning Explainable Rules for Recommendation with Knowledge Graph. In *WWW*. ACM, 1210–1221.

[28] Enrico Palumbo, Giuseppe Rizzo, and Raphaël Troncy. 2017. entity2rec: Learning User-Item Relatedness from Knowledge Graphs for Top-N Item Recommendation. In *RecSys*. ACM, 32–36.

[29] Dean Pomerleau. 1991. Efficient Training of Artificial Neural Networks for Autonomous Navigation. *Neural Computation* 3, 1 (1991), 88–97.

[30] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*. AUAI Press, 452–461.

[31] Wenjie Shang, Yang Yu, Qingyang Li, Zhiwei Qin, Yiping Meng, and Jieping Ye. 2019. Environment Reconstruction with Hidden Confounders for Reinforcement Learning based Recommendation. In *KDD*. ACM, 566–576.

[32] Guy Shani, David Heckerman, and Ronen I. Brafman. 2005. An MDP-Based Recommender System. *J. Mach. Learn. Res.* 6 (2005), 1265–1295.

[33] Amit Sharma and Dan Cosley. 2013. Do social explanations work?: studying and modeling the effects of social explanations in recommender systems. In *WWW*. ACM, 1133–1144.

[34] Yizhou Sun and Jiawei Han. 2012. Mining heterogeneous information networks: a structural analysis approach. *SIGKDD Explorations* 14, 2 (2012), 20–28.

[35] Zhu Sun, Jie Yang, Jie Zhang, Alessandro Bozzon, Long-Kai Huang, and Chi Xu. 2018. Recurrent knowledge graph embedding for effective recommendation. In *RecSys*. ACM, 297–305.

[36] Richard S. Sutton. 1988. Learning to Predict by the Methods of Temporal Differences. *Machine Learning* 3 (1988), 9–44.

[37] Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement learning - an introduction*. MIT Press.

[38] Nava Tintarev and Judith Masthoff. 2007. A survey of explanations in recommender systems. In *ICDE workshop*. IEEE, 801–810.

[39] 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.

[40] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. DKN: Deep Knowledge-Aware Network for News Recommendation. In *WWW*. ACM, 1835–1844.

[41] Lu Wang, Wei Zhang, Xiaofeng He, and Hongyuan Zha. 2018. Supervised Reinforcement Learning with Recurrent Neural Network for Dynamic Treatment Recommendation. In *KDD*. ACM, 2447–2456.

[42] Xiting Wang, Yiru Chen, Jie Yang, Le Wu, Zhengtao Wu, and Xing Xie. 2018. A Reinforcement Learning Framework for Explainable Recommendation. In *ICDM*. IEEE, 587–596.

[43] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable Reasoning over Knowledge Graphs for Recommendation. In *AAAI*. AAAI Press, 5329–5336.

[44] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. In *AAAI*. AAAI Press, 1112–1119.

[45] Ronald J. Williams. 1992. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning* 8 (1992), 229–256.

[46] Yikun Xian, Zuohui Fu, S. Muthukrishnan, Gerard de Melo, and Yongfeng Zhang. 2019. Reinforcement Knowledge Graph Reasoning for Explainable Recommendation. In *SIGIR*. ACM, 285–294.

[47] Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *EMNLP*.

[48] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. 2014. Personalized entity recommendation: a heterogeneous information network approach. In *WSDM*. ACM, 283–292.

[49] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative Knowledge Base Embedding for Recommender Systems. In *KDD*. ACM, 353–362.

[50] Jing Zhang, Bowen Hao, Bo Chen, Cuiping Li, Hong Chen, and Jimeng Sun. 2019. Hierarchical Reinforcement Learning for Course Recommendation in MOOCs. In *AAAI*. AAAI Press, 435–442.

[51] Yongfeng Zhang, Qingyao Ai, Xu Chen, and W. Bruce Croft. 2017. Joint Representation Learning for Top-N Recommendation with Heterogeneous Information Sources. In *CIKM*. ACM, 1449–1458.

[52] 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. In *SIGIR*. 83–92.

[53] Huan Zhao, Quanming Yao, Jianda Li, Yangqiu Song, and Dik Lun Lee. 2017. Meta-Graph Based Recommendation Fusion over Heterogeneous Information Networks. In *KDD*. ACM, 635–644.

[54] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. 2018. Deep reinforcement learning for page-wise recommendations. In *RecSys*. ACM, 95–103.

[55] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. 2018. Recommendations with Negative Feedback via Pairwise Deep Reinforcement Learning. In *KDD*. ACM, 1040–1048.

[56] 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 *WWW*. ACM, 167–176.