Unified Conversational Recommendation Policy Learning via Graph-based Reinforcement Learning

Yang Deng[1], Yaliang Li[2], Fei Sun[2], Bolin Ding[2], Wai Lam[1]
[1]The Chinese University of Hong Kong, [2]Alibaba Group
{ydeng, wlam}@se.cuhk.edu.hk, {yaliang.li, ofey.sf, bolin.ding}@alibaba-inc.com

ABSTRACT

Conversational recommender systems (CRS) enable the traditional recommender systems to explicitly acquire user preferences towards items and attributes through interactive conversations. Reinforcement learning (RL) is widely adopted to learn conversational recommendation policies to decide what attributes to ask, which items to recommend, and when to ask or recommend, at each conversation turn. However, existing methods mainly target at solving one or two of these three decision-making problems in CRS with separated conversation and recommendation components, which restrict the scalability and generality of CRS and fall short of preserving a stable training procedure. In the light of these challenges, we propose to formulate these three decision-making problems in CRS as a unified policy learning task. In order to systematically integrate conversation and recommendation components, we develop a dynamic weighted graph based RL method to learn a policy to select the action at each conversation turn, either asking an attribute or recommending items. Further, to deal with the sample efficiency issue, we propose two action selection strategies for reducing the candidate action space according to the preference and entropy information. Experimental results on two benchmark CRS datasets and a real-world E-Commerce application show that the proposed method not only significantly outperforms state-of-the-art methods but also enhances the scalability and stability of CRS.

CCS CONCEPTS

• **Information systems** → **Users and interactive retrieval**; **Recommender systems**.

KEYWORDS

Conversational Recommendation, Reinforcement Learning, Graph Representation Learning

ACM Reference Format:
Yang Deng, Yaliang Li, Fei Sun, Bolin Ding, Wai Lam. 2021. Unified Conversational Recommendation Policy Learning via Graph-based Reinforcement Learning. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3404835.3462913

SIGIR '21, July 11–15, 2021, Virtual Event, Canada

ACM ISBN 978-1-4503-8037-9/21/07…$15.00
https://doi.org/10.1145/3404835.3462913

1 INTRODUCTION

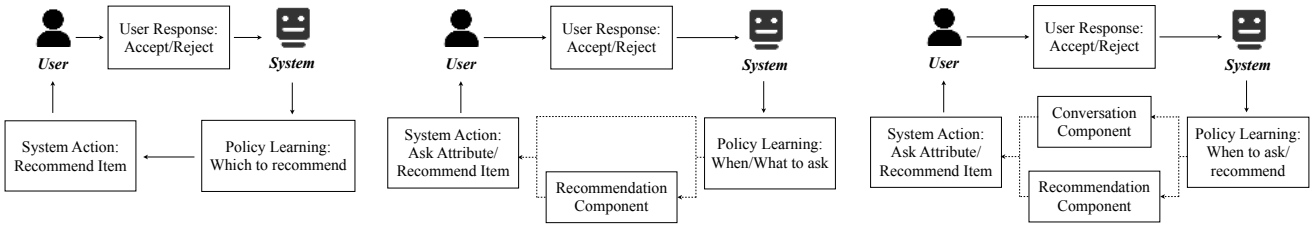
Conversational recommender systems (CRS) aim to learn user's preferences and make recommendations through interactive conversations [12, 16, 41]. Since it has the natural advantage of explicitly acquiring user's preferences and revealing the reasons behind recommendation, CRS has become one of the trending research topics for recommender systems and is gaining increasing attention. Unlike traditional recommender systems [8, 23, 43] or interactive recommender systems (IRS) [49, 51], which mainly focus on solving the problem of which items to recommend, there exists generally the other two core research questions for CRS [11], namely what questions to ask and when to ask or recommend. Recent works have demonstrated the importance of interactivity of asking clarifying questions in CRS [4, 41, 50]. More importantly, deciding when to ask or recommend is the key to coordinating conversation and recommendation for developing an effective CRS [12, 14, 26].

Different problem settings of CRS have been proposed, either from the perspective of dialog systems, being a variation of task-oriented dialog [13, 16, 48], or from the perspective of recommender systems, being an enhanced interactive recommender system [3, 12, 26]. In this work, we study the multi-round conversational recommendation (MCR) setting [12, 26], where the system asks questions about users' preferences on certain attributes or recommends items multiple times, and the goal is to make successful recommendation with the minimum number of interactions.

In MCR scenario, the CRS is typically formulated as a multi-step decision-making process and solved by reinforcement learning (RL) methods for policy learning [12, 14, 26]. As shown in Figure 1(a), RL-based IRS is only required to learn the policy to decide which items to recommend. However, the situation is more complicated in CRS, since there are two components that need to be coherently considered [19], i.e., conversation and recommendation components. Existing methods CRM [26] and EAR [12] employ policy gradient [27] to improve the strategies of when and what attributes to ask, while the recommendation decision is made by an external recommendation model. In order to reduce the action space in policy learning, another state-of-the-art method SCPR [14] only considers learning the policy of when to ask or recommend, while two isolated components are responsible for the decision of what to ask and which to recommend. The policy learning frameworks of these two kinds of CRS are presented in Figure 1(b) and 1(c).

Despite the effectiveness of these methods, there are some issues remained to be tackled for real-world applications: (i) The models trained with existing CRS methods lack generality to different

Work done when Yang Deng was an intern at Alibaba. This work was supported by Alibaba Group through Alibaba Research Intern Program, and a grant from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project Codes: 14200719).



(a) Interactive Recommender System (b) Conversational Recommender System, e.g., CRM, EAR (c) Conversational Recommender System, e.g., SCPR

Figure 1: Illustration of policy learning frameworks for IRS and CRS, including CRM [26], EAR [12], and SCPR [14].

domains or applications, since there are three different decision-making processes to be considered in CRS, including what attributes to ask, which items to recommend, and when to ask or recommend. It requires extra efforts to train an offline recommendation model [12, 14] or pretrain the policy network with synthetic dialogue history [12, 26]. (ii) The policy learning is hard to converge, since the conversation and recommendation components are isolated and lack of mutual influence during the training procedure.

To address these issues, in this work, we formulate the aforementioned three separated decision-making processes in CRS as a unified policy learning problem to harness the ultimate goal of CRS as well as fill the gap between the recommendation and conversation components during the training procedure. Such unified conversational recommendation policy learning (UCRPL) aims at learning a unified policy to decide the action, either ask an attribute or recommend items, at each conversation turn to maximize the cumulative utility over the whole MCR process. The overview of the proposed unified policy learning for CRS is depicted in Figure 2.

However, the UCRPL problem comes along with two challenges: (i) How to systematically combine conversation and recommendation components for unified policy learning? (ii) How to deal with sample efficiency issues? As the action space becomes overwhelmingly large in UCRPL, including all available attributes and items, it requires tremendous amount of interaction data to learn the optimal policy. Fortunately, the graph structure, capturing the rich correlated information among different types of nodes (i.e., users, items, and attributes), enables us to discover collaborative user preferences towards attributes and items. Thus, we can leverage the graph structure to integrate the recommendation and conversation components as an organic whole, where the conversation session can be regarded as a sequence of nodes maintained in the graph to dynamically exploit the conversation history for predicting the action at the next turn. On the other hand, although the connectivity of the graph can also be used to eliminate invalid actions by path reasoning [14], there are still a large number of candidates left for action searching. Since users are not likely to be interested in all items and attributes, we can focus on the potentially important candidates to improve the sample efficiency of UCRPL.

To this end, we propose a novel and adaptive graph-based reinforcement learning framework, namely **UNI**fied **CO**nversational **R**ecomme**N**der (**UNICORN**). Specifically, due to the evolving nature of CRS, we leverage a dynamic weighted graph to model the changing interrelationships among users, items and attributes during the conversation, and consider a graph-based Markov Decision Process (MDP) environment for simultaneously handling the

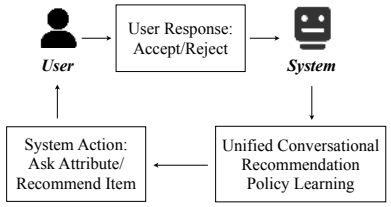


Figure 2: Illustration of unified policy learning for CRS.

decision-making of recommendation and conversation. Then we integrate graph-enhanced representation learning and sequential conversation modeling to capture dynamic user preferences towards items and attributes. In addition, two simple yet effective action selection strategies are designed to handle the sample efficiency issue. Instead of enumerating the whole candidate item and attribute set, we adopt preference-based item selection and weighted entropy-based attribute selection strategies to only consider potentially important actions.

In summary, the contributions of this paper are as follows:

- We formulate three separated decision-making processes in conversational recommender systems, namely when to ask or recommend, what to ask, and which to recommend, as a unified conversational recommendation policy learning problem.
- To tackle the challenges in UCRPL problem, we propose a novel and adaptive reinforcement learning framework, namely **UNI**fied **CO**nversational **R**ecomme**N**der (**UNICORN**), based on dynamic weighted graph. To handle the sample efficiency issue, we further design two simple yet effective action selection strategies.
- Experimental results show that the proposed method significantly outperforms state-of-the-art CRS methods across four public benchmark datasets and a real-world E-Commerce application.

2 RELATED WORKS

Conversational Recommendation. Based on the problem settings, current CRS studies can be categorized into four directions [5, 11]: (1) Exploration-Exploitation Trade-offs for Cold-start Users [4, 17, 40]. These approaches leverage bandit approaches to balance the exploration and exploitation trade-offs for cold-start users in conversational recommendation scenarios. (2) Question-driven Approaches [3, 41, 50] aim at asking questions to users to get more information about their preferences, which is often addressed as "asking clarification/clarifying question". (3) Dialogue Understanding and Generation [16, 18, 47, 48]. These studies focus on how

to understand users' preferences and intentions from their utterances and generate fluent responses so as to deliver natural and effective dialogue actions. (4) Multi-round Conversational Recommendation [12, 14, 26]. Under this problem setting, the system asks questions about the user's preferences or makes recommendations multiple times, with the goal of achieving engaging and successful recommendations with fewer turns of conversations. Among these problem settings, we focus on the MCR problem.

RL in Recommendation. Reinforcement learning (RL) has been widely introduced into recommender systems due to its advantage of considering users' long-term feedbacks [44, 45]. RL-based recommendation formulates the recommendation procedure as an MDP of the interactions between the user and a recommendation agent, and employs RL algorithms to learn the optimal recommendation strategies [22, 25, 44, 45]. Recent works on sequential recommendation [30, 37] and interactive recommendation [39, 49, 51] adopt RL to capture users' dynamic preferences for generating accurate recommendations over time. The goal of these approaches typically is to learn an effective policy for determining which items to recommend. As for CRS, RL-based methods are adopted to improve the strategies of the other two decision processes, including (i) what attributes to ask [12, 26] and (ii) when to ask or recommend [14]. In order to simplify the overall framework of MCR with better scalability and generality, we formulate these three core decision processes in CRS as a unified policy learning problem.

Graph-based Recommendation. Graph-based recommendation studies mainly leverage the graph structure for two purposes. The first one is to enhance the recommendation performance by graph-based representation learning, including exploiting the structure information for collaborative filtering [7, 31, 46], and adopting knowledge graph embeddings as rich context information [9, 38]. The other group of studies models recommendation as a path reasoning problem for building explainable recommender systems [20, 32]. Recent years have witnessed many successful applications of graph-based RL methods on different scenarios of recommender systems [14, 15, 33, 36, 42, 49]. For example, Xian et al. [36] employ a policy-guided graph search method to sample reasoning paths for recommendation, which is enhanced with adversarial actor-critic for demonstration-guided path reasoning [42]. Lei et al. [15] and Zhou et al. [49] employ graph convolutional network (GCN) [10] for state representation learning to enhance the performance of traditional RL methods on recommendation policy learning. In this work, we study graph-based RL method for the conversational recommendation scenario based on a dynamic weighted graph.

3 PROBLEM DEFINITION

Multi-round Conversational Recommendation. In this work, we focus on the multi-round conversational recommendation (MCR) scenario [12, 14], the most realistic conversational recommendation setting proposed so far, in which the CRS is able to ask questions about attributes or make recommendations multiple times.

Specifically, on the system side, the CRS maintains a large set of items $\mathcal{V}$ to be recommended, and each item v is associated with a set of attributes $\mathcal{P}_v$. In each episode, a conversation session is initialized by a user u specifying an attribute p_0. Then, the CRS is free to ask the user's preference on an attribute selected from the candidate attribute set $\mathcal{P}_{\text{cand}}$ or recommend a certain number of items (e.g., top-K) from the candidate item set $\mathcal{V}_{\text{cand}}$. Following the assumptions from Lei et al. [12], the user preserves clear preferences towards all the attributes and items. Thus, the user will respond accordingly, either accepting or rejecting the asked attributes or the recommended items. The CRS updates the candidate attribute and item sets, and decides the next action based on the user response. The system-ask and user-respond process repeats until the CRS hits the target item or reaches the maximum number of turn T.

Unified Conversational Recommendation Policy Learning. Multi-round conversational recommendation aims to make successful recommendations within a multi-round conversation session with the user. At each timestep t, according to the observation on past interactions, the CRS selects an action a_t, either asking an attribute or recommending items. In return, the user expresses his/her feedback (accept or reject). This process repeats until the CRS hits the user-preferred items or reaches the maximum number of turn T. Such MCR task can be formulated as a Markov Decision Process (MDP). The goal of the CRS is to learn a policy π maximizing the expected cumulative rewards over the observed MCR episodes as

$$\pi^* = \arg\max_{\pi \in \Pi} \mathbb{E}\left[\sum_{t=0}^{T} r(s_t, a_t)\right], \tag{1}$$

where s_t is the state representation learned from the system state and the conversation history, a_t is the action that the agent takes at timestep t, and $r(\cdot)$ is the intermediate reward, abbreviated as r_t.

4 METHODOLOGY

The overview of the proposed method, UNICORN, is depicted in Figure 3, which consists of four main components: Graph-based MDP Environment, Graph-enhanced State Representation Learning, Action Selection Strategy, and Deep Q-Learning Network.

4.1 Graph-based MDP Environment

The MDP environment is responsible for informing the agent about the current state and possible actions to take, and then rewards the agent based on how the current policy fits the observed user interactions. Formally, the MDP environment can be defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the action space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ refers to the state transition function, and $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function.

4.1.1 ***State.*** As for the graph-based MDP environment, the state $s_t \in \mathcal{S}$ at timestep t is supposed to contain all the given information for conversational recommendation, including the previous conversation history and the full graph $\mathcal{G}$ that includes all the users, items, and attributes. Given a user u, we consider two major elements:

$$s_t = [\mathcal{H}_u^{(t)}, \mathcal{G}_u^{(t)}], \tag{2}$$

where $\mathcal{H}_u^{(t)} = [\mathcal{P}_u^{(t)}, \mathcal{P}_{\text{rej}}^{(t)}, \mathcal{V}_{\text{rej}}^{(t)}]$ denotes the conversation history until timestep t, and $\mathcal{G}_u^{(t)}$ denotes the dynamic subgraph of $\mathcal{G}$ for the user u at timestep t (The graph construction will be introduced in Section 4.2). $\mathcal{P}_u$ denotes the user-preferred attribute. $\mathcal{P}_{\text{rej}}$ and $\mathcal{V}_{\text{rej}}$ are the attributes and items rejected by the user, respectively. The initial state s_0 is initialized by the user-specified attribute p_0, i.e., $s_0 = [[\{p_0\}, \{\}, \{\}], \mathcal{G}_u^{(0)}]$.

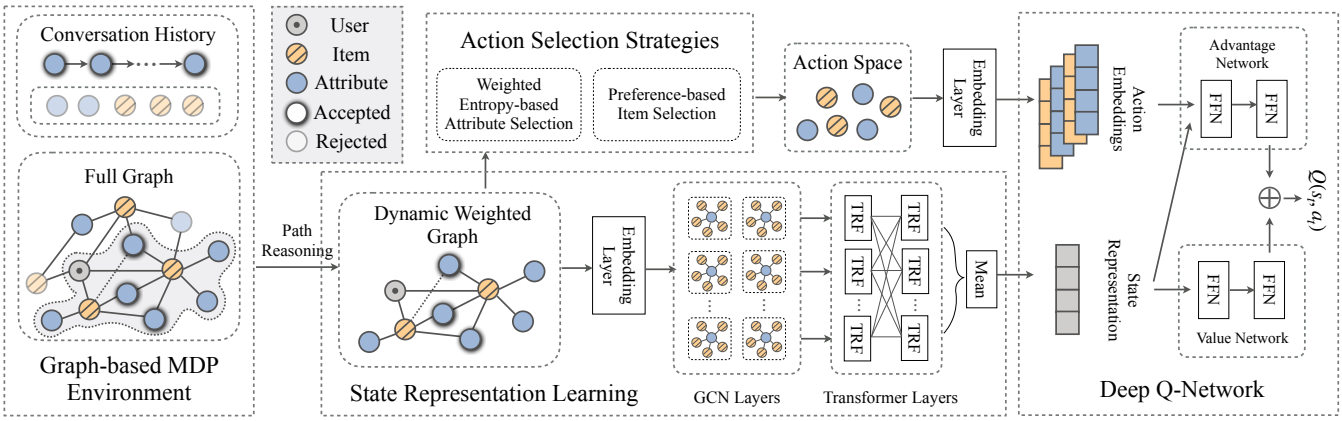


Figure 3: Overview of the proposed method, UNICORN, for unified conversational recommendation policy learning.

4.1.2 ***Action.*** According to the state s_t, the agent takes an action $a_t \in \mathcal{A}$, where a_t can be selected from the candidate item set $\mathcal{V}_{\text{cand}}^{(t)}$ to recommend items or from the candidate attribute set $\mathcal{P}_{\text{cand}}^{(t)}$ to ask attributes. Following the path reasoning approach [14], we have

$$\mathcal{V}_{\text{cand}}^{(t)} = \mathcal{V}_{\mathcal{P}_u^{(t)}} \setminus \mathcal{V}_{\text{rej}}^{(t)}, \quad \mathcal{P}_{\text{cand}}^{(t)} = \mathcal{P}_{\mathcal{V}_{\text{cand}}^{(t)}} \setminus (\mathcal{P}_u^{(t)} \cup \mathcal{P}_{\text{rej}}^{(t)}), \tag{3}$$

where $\mathcal{V}_{\mathcal{P}_u^{(t)}}$ is the set of item vertices directly connecting all $\mathcal{P}_u^{(t)}$ (i.e., items satisfying all the preferred attributes), and $\mathcal{P}_{\mathcal{V}_{\text{cand}}^{(t)}}$ is the set of attribute vertices directly connecting to one of $\mathcal{V}_{\text{cand}}^{(t)}$ (i.e., attributes belonging to at least one of the candidate items).

4.1.3 ***Transition.*** We consider that the current state s_t will transition to the next state s_{t+1} when the user responds to the action a_t. In specific, if CRS asks an attribute p_t and the user accepts it, the next state s_{t+1} will be updated by $\mathcal{P}_u^{(t+1)} = \mathcal{P}_u^{(t)} \cup p_t$. Conversely, if the user rejects the action a_t, s_{t+1} will be updated by $\mathcal{P}_{\text{rej}}^{(t+1)} = \mathcal{P}_{\text{rej}}^{(t)} \cup a_t$ or $\mathcal{V}_{\text{rej}}^{(t+1)} = \mathcal{V}_{\text{rej}}^{(t)} \cup a_t$ for $a_t \in \mathcal{P}$ or $a_t \in \mathcal{V}$, respectively. As a result, the next state s_{t+1} will be $[\mathcal{H}_u^{(t+1)}, \mathcal{G}_u^{(t+1)}]$.

4.1.4 ***Reward.*** Following previous MCR studies [12, 14], our environment contains five kinds of rewards, namely, (1) $r_{\text{rec\_suc}}$, a strongly positive reward when the user accepts the recommended items, (2) $r_{\text{rec\_fail}}$, a negative reward when the user rejects the recommended items, (3) $r_{\text{ask\_suc}}$, a slightly positive reward when the user accepts the asked attribute, (4) $r_{\text{ask\_fail}}$, a negative reward when the user rejects the asked attribute, and (5) r_{quit}, a strongly negative reward when reaching the maximum number of turns.

4.2 Graph-enhanced State Representation

As we formulate conversational recommendation as a unified policy learning problem over a graph-based MDP environment, it is required to encode both the conversational and graph structural information into the latent distributed representations. In order to make use of the interrelationships among users, items, and attributes, we first adopt graph-based pre-training methods [2, 35] to obtained node embeddings for all the nodes in the full graph $\mathcal{G}$.

4.2.1 ***Dynamic Weighted Graph Construction.*** As shown in Figure 3, we represent the current state of the graph-based MDP environment as a dynamic weighted graph. Formally, we denote an undirected weighted graph as $\mathcal{G} = (\mathcal{N}, \boldsymbol{A})$, with the node $n_i \in \mathcal{N}$, the adjacency matrix element $\boldsymbol{A}_{i,j}$ denoting the weighted edges between nodes n_i and n_j. In our case, given the user u, we denote the dynamic graph at timestep t as $\mathcal{G}_u^{(t)} = (\mathcal{N}^{(t)}, \boldsymbol{A}^{(t)})$:

$$\mathcal{N}^{(t)} = \{u\} \cup \mathcal{P}_u^{(t)} \cup \mathcal{P}_{\text{cand}}^{(t)} \cup \mathcal{V}_{\text{cand}}^{(t)} \tag{4}$$

$$\boldsymbol{A}_{i,j}^{(t)} = \begin{cases} w_v^{(t)}, & \text{if } n_i = u, n_j \in \mathcal{V} \\ 1, & \text{if } n_i \in \mathcal{V}, n_j \in \mathcal{P} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

where $w_v^{(t)}$ is a scalar indicating the recommendation score of the item v in the current state. In order to incorporate the user preference as well as the correlation between the asked attributes and the items, such weight $w_v^{(t)}$ is calculated as

$$w_v^{(t)} = \sigma\Big(\boldsymbol{e}_u^\top \boldsymbol{e}_v + \sum\nolimits_{p \in \mathcal{P}_u^{(t)}} \boldsymbol{e}_v^\top \boldsymbol{e}_p - \sum\nolimits_{p \in \mathcal{P}_{\text{rej}}^{(t)} \cap \mathcal{P}_v^{(t)}} \boldsymbol{e}_v^\top \boldsymbol{e}_p\Big), \tag{6}$$

where $\sigma(\cdot)$ denotes the sigmoid function, $\boldsymbol{e}_u$, $\boldsymbol{e}_v$, and $\boldsymbol{e}_p$ are the embeddings of the user, item, and attribute, respectively.

4.2.2 ***Graph-based Representation Learning.*** In order to comprehensively take advantage of the correlation information among the involved user, items, and attributes from the connectivity of the graph, we employ a graph convolutional network (GCN) [10] to refine the node representations with structural and relational knowledge. The representations of the node n_i in the $(l+1)$-th layer can be computed by:

$$\boldsymbol{e}_i^{(l+1)} = \text{ReLU}\left(\sum\nolimits_{j \in \mathcal{N}_i} \boldsymbol{\Lambda}_{i,j} \boldsymbol{W}_l \boldsymbol{e}_j^{(l)} + \boldsymbol{B}_l \boldsymbol{e}_i^{(l)}\right), \tag{7}$$

where $\mathcal{N}_i$ denotes the neighboring indices of the node n_i, $\boldsymbol{W}_l$ and $\boldsymbol{B}_l$ are trainable parameters representing the transformation from neighboring nodes and the node n_i itself, and $\boldsymbol{\Lambda}$ is a normalization adjacent matrix as $\boldsymbol{\Lambda} = \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{A} \boldsymbol{D}^{-\frac{1}{2}}$ with $\boldsymbol{D}_{ii} = \sum_j \boldsymbol{A}_{i,j}$.

4.2.3 ***Sequential Representation Learning.*** Apart from the interrelationships among the involved user, items, and attributes, the CRS is also expected to model the conversation history in the current state. Unlike previous studies [12, 14] that adopt heuristic features for conversation history modeling, we employ Transformer

encoder [29] for capturing the sequential information of the conversation history as well as attending the important information for deciding the next action. As described in [29], each Transformer layer consists of three components: (i) The layer normalization is defined as LayerNorm$(\cdot)$. (ii) The multi-head attention is defined as MultiHead$(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})$, where $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ are query, key, and value, respectively. (iii) The feed-forward network with ReLU activation is defined as FFN$(\cdot)$. Take the l-th layer for example:

$$\boldsymbol{X}^* = \text{MultiHead}(\boldsymbol{X}^{(l)}, \boldsymbol{X}^{(l)}, \boldsymbol{X}^{(l)}), \tag{8}$$

$$\boldsymbol{X}^{(l+1)} = \text{LayerNorm}(\text{FFN}(\boldsymbol{X}^*) + \boldsymbol{X}^{(l)}), \tag{9}$$

where $\boldsymbol{X} \in \mathbb{R}^{L \times d}$ denotes the embeddings, and L is the sequence length. In our case, the input sequence $\boldsymbol{X}^{(0)}$ is the accepted attributes $\mathcal{P}_u^{(t)}$ in the current conversation history with the learned graph-based representation $\{e_p^{(L_g)} : p \in \mathcal{P}_u^{(t)}\}$, where L_g is the number of layers in GCN. After the sequential learning with L_s Transformer layers, we can aggregate the information learned from both the graph and the conversation history by a mean pooling layer to obtain the state representation of s_t:

$$f_{\theta_S}(s_t) = \text{MeanPool}(\boldsymbol{X}^{L_s}). \tag{10}$$

For simplicity, we denote the learned state representation of s_t as $f_{\theta_S}(s_t)$, where θ_S is the set of all network parameters for state representation learning, including GCN and Transformer layers.

4.3 Action Selection Strategy

A large action search space will harm the performance of the policy learning to a great extent [14]. Thus, it attaches great importance to handle the overwhelmingly large action space in UCRPL. To this end, we propose two simple strategies to improve the sample efficiency for candidate action selection.

Preference-based Item Selection. In general, for candidate items to be recommended, we can consider only the action of making recommendations from a small number of candidate items that fit the user preference the most, since users are not likely to be interested in all items. To achieve this, we select top-K_v candidate items from $\mathcal{V}_{\text{cand}}^{(t)}$ into the candidate action space $\mathcal{A}_t$ at each timestep t, which is ranked by the recommendation score $w_v^{(t)}$ in Eq.(6).

Weighted Entropy-based Attribute Selection. Whereas for candidate attributes to be asked, the expected one is supposed to be able to not only better eliminate the uncertainty of candidate items, but also encode the user preference. Inspired by [14], we adopt weighted entropy as the criteria to prune candidate attributes:

$$w_p^{(t)} = -\text{prob}(p^{(t)}) \cdot \log(\text{prob}(p^{(t)})), \tag{11}$$

$$\text{prob}(p^{(t)}) = \sum\nolimits_{v \in \mathcal{V}_{\text{cand}}^{(t)} \cap \mathcal{V}_p^{(t)}} w_v^{(t)} \Big/ \sum\nolimits_{v \in \mathcal{V}_{\text{cand}}^{(t)}} w_v^{(t)}, \tag{12}$$

where $\mathcal{V}_p$ denotes the items that have the attribute p. Similar to item selection, we also select top-K_p candidate attributes from $\mathcal{P}_{\text{cand}}^{(t)}$ into $\mathcal{A}_t$ based on the weighted entropy score $w_p^{(t)}$.

4.4 Deep Q-Learning Network

After obtaining the graph-enhanced state representation and the candidate action space, we introduce the deep Q-learning network (DQN) [21] to conduct the unified conversational recommendation policy learning. We further implement some techniques to enhance and stabilize the training of DQN. The training procedure of the Unified Conversational Recommender is presented in Algorithm 1.

Algorithm 1: Unified Conversational Recommender

Input: $\{\mathbf{e}_i\}_{i \in \mathcal{N}}$; $\mathcal{D}$; τ; ϵ; γ; K; T; K_v; K_p;
Output: θ_S; θ_Q;

Initialize all parameters: θ_S, θ_Q, $\theta'_Q \leftarrow \theta_Q$;
for *episode* $= 1, 2, \ldots, N$ **do**
  User u starts the conversation by specifying an attribute p_0;
  Update: $\mathcal{P}_u^{(0)} = \{p_0\}$, $\mathcal{P}_{\text{rej}}^{(0)} = \{\}$, $\mathcal{V}_{\text{rej}}^{(0)} = \{\}$, $\mathcal{H}_u^{(0)} = [\mathcal{P}_u^{(0)}, \mathcal{P}_{\text{rej}}^{(0)}, \mathcal{V}_{\text{rej}}^{(0)}]$, $s_0 = [\mathcal{H}_u^{(0)}, \mathcal{G}_u^{(0)}]$;
  Get candidate action space $\mathcal{A}_0$ via Action Selection;
  for *turn* $t = 0, 1, \ldots, T-1$ **do**
    Get state representation $f_{\theta_S}(s_t)$ via Eq.(10);
    Select an action a_t by ϵ-greedy w.r.t Eq.(13);
    Receive reward r_t;
    Update the next state $s_{t+1} = \mathcal{T}(s_t, a_t)$;
    Get $\mathcal{A}_{t+1}$ via Action Selection;
    Store $(s_t, a_t, r_t, s_{t+1}, \mathcal{A}_{t+1})$ to buffer $\mathcal{D}$;
    Sample mini-batch of $(s_t, a_t, r_t, s_{t+1}, \mathcal{A}_{t+1})$ w.r.t Eq.(19);
    Compute the target value y_t via Eq. (17);
    Update θ_S, θ_Q via SGD w.r.t the loss function Eq.(15);
    Update θ'_Q via Eq.(18) ;
  end
end

4.4.1 ***Dueling Q-Network***. Following the standard assumption that delayed rewards are discounted by a factor of γ per timestep, we define the Q-value $Q(s_t, a_t)$ as the expected reward based on the state s_t and the action a_t. As shown in the rightmost part of Figure 3, the dueling Q-network [34] employs two deep neural networks to compute the value function $f_{\theta_V}(\cdot)$ and advantage function $f_{\theta_A}(\cdot)$, respectively. Then the Q-function can be calculated by:

$$Q(s_t, a_t) = f_{\theta_V}(a_t) + f_{\theta_A}(f_{\theta_S}(s_t), a_t), \tag{13}$$

where $f_{\theta_V}(\cdot)$ and $f_{\theta_A}(\cdot)$ are two separate multi-layer perceptions with parameters θ_V and θ_A, respectively, and let $\theta_Q = \{\theta_V, \theta_A\}$.

The optimal Q-function $Q^*(s_t, a_t)$, which has the maximum expected reward achievable by the optimal policy π^*, follows the Bellman equation [1] as:

$$Q^*(s_t, a_t) = \mathbb{E}_{s_{t+1}}\Big[r_t + \gamma \max_{a_{t+1} \in \mathcal{A}_{t+1}} Q^*(s_{t+1}, a_{t+1}) | s_t, a_t\Big]. \tag{14}$$

4.4.2 ***Double Q-Learning with Prioritized Experience Replay***. During each episode in the MCR process, at each timestep t, the CRS agent obtains the current state representation $f_{\theta_S}(s_t)$ via the graph-enhanced state representational learning described in Section 4.2. Then the agent selects an action a_t from the candidate action space $\mathcal{A}_t$, which is obtained via the action selection strategies described in Section 4.3. Here we incorporate ϵ-greedy method to balance the exploration and exploitation in action sampling (i.e., select a greedy action based on the max Q-value with probability $1 - \epsilon$, and a random action with probability ϵ).

Then, the agent will receive the reward r_t from the user's feedback. According to the feedback, the current state s_t transitions to

the next state s_{t+1}, and the candidate action space $\mathcal{A}_{t+1}$ is updated accordingly. The experience $(s_t, a_t, r_t, s_{t+1}, \mathcal{A}_{t+1})$ is then stored into the replay buffer $\mathcal{D}$. To train DQN, we sample mini-batch of experiences from $\mathcal{D}$, and minimize the following loss function:

$$\mathcal{L}(\theta_Q, \theta_S) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}, \mathcal{A}_{t+1}) \sim \mathcal{D}} \left[(y_t - Q(s_t, a_t; \theta_Q, \theta_S))^2 \right], \quad (15)$$

$$y_t = r_t + \gamma \max_{a_{t+1} \in \mathcal{A}_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_Q, \theta_S), \quad (16)$$

where y_t is the target value based on the currently optimal Q^*.

To alleviate the overestimation bias problem in conventional DQN, we adopt Double Q-learning [28], which employs a target network Q' as a periodic copy from the online network. The target value of the online network is then changed to:

$$y_t = r_t + \gamma Q'\big(s_{t+1}, \operatorname*{arg\,max}_{a_{t+1} \in \mathcal{A}_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_Q, \theta_S); \theta_{Q'}, \theta_S\big), \quad (17)$$

where $\theta_{Q'}$ denotes the parameter of the target network, which is updated by the soft assignment as:

$$\theta_{Q'} = \tau \theta_Q + (1 - \tau)\theta_{Q'}, \quad (18)$$

where τ is the update frequency.

In addition, the conventional DQN samples uniformly from the replay buffer. In order to sample more frequently those important transitions from which there is much to learn, we employ prioritized replay [24] as a proxy for learning potential, which samples transitions with probability δ relative to the absolute TD error:

$$\delta \propto \left| r_{t+1} + \gamma Q'\big(s_{t+1}, \operatorname*{arg\,max}_{a_{t+1} \in \mathcal{A}_{t+1}} Q(s_{t+1}, a_{t+1})\big) - Q(s_t, a_t) \right|. \quad (19)$$

4.4.3 ***Model Inference.*** With the learned UNICORN model, given a user and his/her conversation history, we follow the same process to obtain the candidate action space and the current state representation, and then decide the next action according to max Q-value in Eq.(13). If the selected action points to an attribute, the system will ask the user's preference on the attribute. Otherwise, the system will recommend top-K items with the highest Q-value to the user.

5 EXPERIMENT

5.1 Dataset

We evaluate the proposed method, UNICORN, on four existing multi-round conversational recommendation benchmark datasets, and further conduct evaluation on a real-world E-Commerce platform. The statistics of these datasets are presented in Table 1.

- **LastFM** and **Yelp**. The LastFM dataset is used for evaluation on music artist recommendation, while the Yelp dataset is for business recommendation. Lei et al. [12] manually categorize the original attributes in LastFM into 33 coarse-grained groups, and build a 2-layer taxonomy with 29 first-layer categories for Yelp.
- **LastFM*** and **Yelp***. Lei et al. [14] consider that it is not realistic to manually merge attributes for practical applications, so they adopt original attributes to reconstruct these two datasets. For a fair comparison, we adopt both versions for experiments.
- **E-Commerce**. In order to evaluate the proposed method in real-world E-Commerce scenario, we collect the dataset by sampling logs from Taobao, the largest E-Commerce platform in China, in the same way as LastFM and Yelp. We randomly sample 30,000 items from the Women_Dress category and a certain number of users who have interacted with these items. And we treat the concatenation of the product property and the property value as the attribute to be asked, e.g., "Color=Red", where "Color" is a product property and "Red" is a property value. Following the MCR data construction described in Lei et al. [12], we obtain the E-Commerce dataset with the statistics shown in Table 1. As for graph construction, we consider two kinds of relations, including "purchased_by (item→ user)" and "belong_to (attribute→ item)". Compared with the imbalanced ratio of #items:#attributes at LastFM (0.88:1) and Yelp (119:1), the E-Commerce dataset preserves a large number of items as well as attributes, which is more in line with real-world applications in E-Commerce.

Table 1: Summary statistics of datasets.

	LastFM	LastFM*	Yelp	Yelp*	E-Com.
#Users	1,801	1,801	27,675	27,675	26,430
#Items	7,432	7,432	70,311	70,311	29,428
#Interactions	76,693	76,693	1,368,606	1,368,606	748,533
#Attributes	33	8,438	29	590	1,413
#Entities	9,266	17,671	98,605	98,576	57,271
#Relations	4	4	3	3	2
#Triplets	138,215	228,217	2,884,567	2,533,827	2,024,962

5.2 Experimental Settings

5.2.1 ***User Simulator.*** Due to the interactive nature of MCR, it needs to be trained and evaluated by interacting with users. Following the user simulator adopted in [12, 26], we simulate a conversation session for each observed user-item interaction pair (u, v). We regard the item v as the ground-truth target item and treat its attribute set $\mathcal{P}_v$ as the oracle set of attributes preferred by the user u in this conversation session. The session is initialized by the simulated user who specifies a certain attribute randomly chosen from $\mathcal{P}_v$. Then the session follows the process of "System Ask, User Respond" [41] as described in Section 4.

5.2.2 ***Baselines.*** We compare the proposed method with several state-of-the-art methods on MCR as follows:

- **Max Entropy**. This is a rule-based strategy to decide the next action, where the CRS selects an attribute to ask based on the maximum entropy within the current state, or chooses to recommend the top-ranked items with a certain probability [12].
- **Abs Greedy** [4]. This method only performs recommendation actions and updates itself until the CRS makes a successful recommendation or exceeds the maximum turns of conversation.
- **CRM** [26]. This method is originally designed for single-round conversational recommendation, which employs policy gradient [27] to learn the policy deciding when and which attributes to ask. We follow [12] to adapt this method into MCR scenario.
- **EAR** [12]. A three-stage method is proposed to enhance the interaction between the conversation and recommendation components with a similar RL framework as CRM.
- **SCPR** [14]. This is the state-of-the-art method on MCR setting, which leverages interactive path reasoning on the graph to prune off candidate attributes and adopts the DQN [21] framework to determine when to ask or recommend.

Table 2: Experimental Results. † indicates statistically significant improvement ($p < 0.01$) over all baselines. hDCG stands for hDCG@(15,10). SR and hDCG are the higher the better, while AT is the lower the better.

	LastFM			LastFM*			Yelp			Yelp*			E-Commerce		
	SR@15	AT	hDCG	SR@15	AT	hDCG	SR@15	AT	hDCG	SR@15	AT	hDCG	SR@15	AT	hDCG
Abs Greedy	0.222	13.48	0.073	0.635	8.66	0.267	0.264	12.57	0.145	0.189	13.43	0.089	0.273	12.19	0.138
Max Entropy	0.283	13.91	0.083	0.669	9.33	0.269	0.921	6.59	0.338	0.398	13.42	0.121	0.328	12.98	0.112
CRM [26]	0.325	13.75	0.092	0.580	10.79	0.224	0.923	6.25	0.353	0.177	13.69	0.070	0.294	12.11	0.146
EAR [12]	0.429	12.88	0.136	0.595	10.51	0.230	0.967	5.74	0.378	0.182	13.63	0.079	0.381	11.48	0.161
SCPR [14]	0.465	12.86	0.139	0.709	8.43	0.317	0.973	5.67	0.382	0.489	12.62	0.159	0.518	12.32	0.168
UNICORN	**0.535**†	**11.82**†	**0.175**†	**0.788**†	**7.58**†	**0.349**†	**0.985**†	**5.33**†	**0.397**†	**0.520**†	**11.31**†	**0.203**†	**0.602**†	**10.45**†	**0.217**†

5.2.3 ***Evaluation Metrics.*** Following previous studies on multi-round conversational recommendation [12, 14], we adopt success rate at the turn t (SR@t) [26] to measure the cumulative ratio of successful conversational recommendation by the turn t. Besides, average turn (AT) is adopted to evaluate the average number of turns for all sessions (if the conversation session reaches the maximum turn T, the turn for such session is also counted as T). The higher SR@t indicates a better performance of the CRS at a turn t, while the lower AT means an overall higher efficiency.

Although SR@t and AT are widely adopted for the evaluation of CRS, both of them are not sensitive to the rank order in the recommendation results and only consider a certain aspect of the performance of CRS. In order to conduct a comprehensive evaluation of CRS, we propose to extend the normalized discounted cumulative gain (NDCG@K) to be a two-level hierarchical version, namely **hNDCG@(T, K)**, with the definition as follows:

$$hDCG@(T,K) = \sum_{t}^{T}\sum_{k}^{K} r(t,k)\left[\frac{1}{\log_2(t+2)} + \left(\frac{1}{\log_2(t+1)} - \frac{1}{\log_2(t+2)}\right)\frac{1}{\log_2(k+1)}\right], \tag{20}$$

where T and K represent the number of conversation turns and recommended items in each turn, $r(t,k)$ denotes the relevance of the result at the turn t and position k. Then hNDCG@(T, K) can be drawn from hDCG@(T, K) with the same way as the original NDCG@K. Since there is only one target item for each session in MCR, we simply use hDCG@(T, K) for evaluation. The intuition behind hNDCG@(T, K) is that the less number of turns of a successful session is favorable for the CRS, while the target item is expected to be ranked higher in the recommendation list at the success turn.

5.2.4 ***Implementation Details.*** Following [14], we split the E-Commerce dataset by 7:1.5:1.5 for training, validation, and testing, and set the size K of the recommendation list as 10, the maximum turn T as 15. We adopt TransE [2] from OpenKE [6] to pretrain the node embeddings in the constructed graph with the training set. We adopt the user simulator described in Section 5.2.1 to interact with the CRS for online training the model using the validation set. For all implemented methods, we conduct the online training for 10,000 episodes. To maintain a fair comparison with other baselines [12, 14], we adopt the same reward settings to train the proposed model: $r_{\text{rec\_suc}}$=1, $r_{\text{rec\_fail}}$=-0.1, $r_{\text{ask\_suc}}$=0.01, $r_{\text{ask\_fail}}$=-0.1, r_{quit}=-0.3. The hyper-parameters are empirically set as follows: The embedding size and the hidden size are set to be 64 and 100. The numbers of

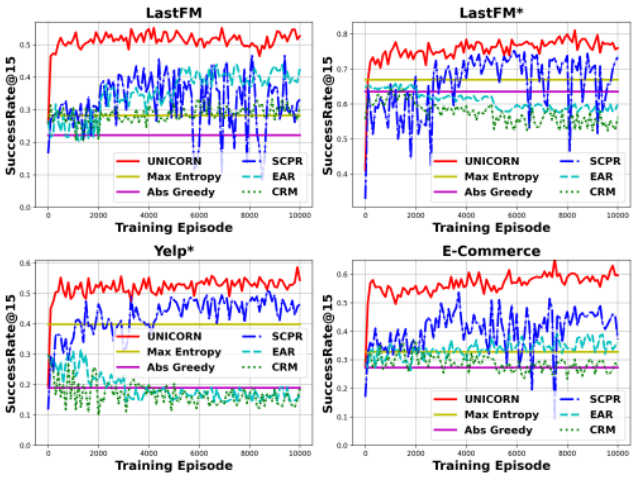


Figure 4: Test Performance at Different Training Episodes.

GCN layers L_g and Transformer layers L_s are set to be 2 and 1, respectively. The numbers of selected candidate attributes K_p and items K_v are set to be 10. During the training procedure of DQN, the size of experience replay buffer is 50,000, and the size of mini-batch is 128. The learning rate and the L_2 norm regularization are set to be 1e-4 and 1e-6, with Adam optimizer. The discount factor γ and the update frequency τ are set to be 0.999 and 0.01.

5.3 Performance Comparison

5.3.1 ***Overall Performance.*** Table 2 shows the performance comparison between the proposed method, UNICORN, and all baselines across five datasets. In general, UNICORN outperforms all the baselines by achieving a significantly higher success rate and less average turn, which is also comprehensively validated by the improvements on hDCG. As for the real-world E-Commerce dataset, SCPR outperforms EAR and CRM, whose performance is largely affected by the large action space in E-Commerce dataset. Despite the effectiveness of SCPR to handle the issue of large action space in CRS, its performance in such a real-world application is still restricted by the separation of its conversation and recommendation components. Our UNICORN not only enables the conversation and recommendation to be mutually enhanced during the training process, but also attains an effective sample efficiency with the proposed action selection strategies. This leads to a substantial margin

Table 3: Ablation Study. SR and hDCG are the higher the better, while AT is the lower the better.

	LastFM			LastFM*			Yelp			Yelp*			E-Commerce		
	SR@15	AT	hDCG	SR@15	AT	hDCG	SR@15	AT	hDCG	SR@15	AT	hDCG	SR@15	AT	hDCG
UNICORN	**0.535**	**11.82**	**0.175**	**0.788**	**7.58**	**0.349**	**0.985**	**5.33**	**0.397**	**0.520**	**11.31**	**0.203**	**0.602**	**10.45**	**0.217**
(a) - w/o Transformer	0.478	12.17	0.147	0.733	7.78	0.320	0.958	6.18	0.370	0.470	11.62	0.167	0.545	11.23	0.178
(b) - w/o GCN	0.440	12.74	0.139	0.744	7.62	0.327	0.978	5.48	0.394	0.481	11.45	0.171	0.554	11.07	0.183
(c) - Random Embeddings	0.310	13.31	0.096	0.665	9.24	0.291	0.905	7.66	0.318	0.196	13.95	0.049	0.220	13.55	0.069
(d) - FM Embeddings	0.465	12.26	0.144	0.748	7.68	0.328	0.982	5.50	0.394	0.496	12.25	0.161	0.579	10.89	0.202
(e) - Preference-based Attr. Sel.	0.467	12.33	0.129	0.712	8.24	0.303	0.978	5.74	0.385	0.489	11.60	0.171	0.569	11.06	0.206
(f) - Entropy-based Attr. Sel.	0.507	11.96	0.165	0.756	7.63	0.324	0.978	5.71	0.388	0.502	11.33	0.192	0.585	10.69	0.214
(g) - w/o Attribute Selection	0.487	12.12	0.151	0.604	9.88	0.258	0.942	6.80	0.357	0.220	12.97	0.067	0.434	12.01	0.127
(h) - w/o Item Selection	0.150	13.83	0.055	0.638	8.85	0.294	0.780	9.16	0.276	0.158	13.60	0.054	0.144	13.85	0.056

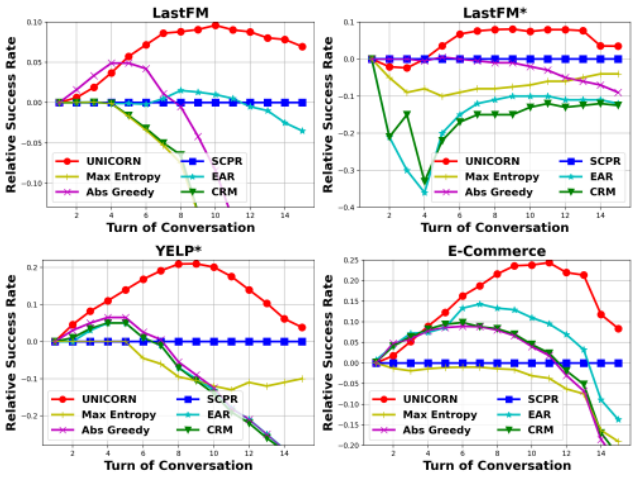


Figure 5: Comparisons at Different Conversation Turns.

from these baselines, about 18% for SR@15, 2 turns for AT, and 30% for hDCG. Detailed analyses can be found as follows.

5.3.2 ***Training Efficiency.*** Figure 4 shows the test performance curves of different methods. Since Max Entropy and Abs Greedy are unsupervised methods, their curves are presented as two lines for comparisons. It can be clearly observed that UNICORN is trained more stably and requires fewer training episodes (i.e., interaction data) to achieve a better performance than other strong baselines. Among these baselines, the curve of SCPR is the most vibrant, since it only considers the policy of when to ask or recommend, while the decisions of question-asking and recommendation are made by two separated components. As for EAR and CRM, due to the large action space in the last three datasets, there is no much performance increase during the online training process, even getting worse. These results demonstrate the efficiency and effectiveness of the proposed unified policy learning for CRS.

5.3.3 ***Comparison at Different Conversation Turns.*** Besides SR@15, we also present the performance comparison of success rate at each turn (SR@t) in Figure 5. In order to better observe the differences among different methods, we report the relative success rate compared with the state-of-the-art baseline SCPR. For example, the line of $y = 0$ represents the curve of Success Rate* for SCPR against itself. There are several notable observations as follows:

(i) The proposed UNICORN substantially and consistently outperforms these baselines across all the datasets and almost each turn in the conversation session.

(ii) Due to the nature of greedy recommendation approach (Abs Greedy), it may successfully hit target items at the early stage of the conversation, leading to a relatively strong performance at the first few turns, but the performance falls quickly as the turn increases.

(iii) UNICORN achieves an outstanding performance in the middle stage of the conversation, where there are still a large number of candidate items and attributes to be pruned. This shows the strong scalability of UNICORN to effectively handle large candidate action space in different situations.

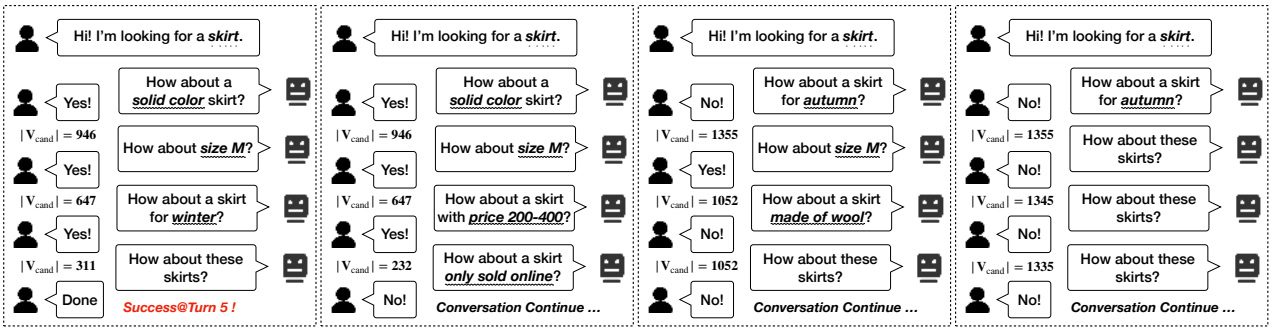
(iv) The performance of SCPR is getting closer to UNICORN at the latter stage of the conversation, as the candidate item and attribute set is getting smaller and the task becomes easier.

(v) EAR and CRM share similar performance as Abs Greedy in those datasets with a large candidate attribute set, i.e., Yelp* and E-Commerce, indicating their policy learning is merely working when encountering a large action space.

5.4 Ablation Study

5.4.1 ***Model Components.*** We first evaluate the model components for the state representation learning, including the graph-based representation learning and the sequential representation learning. The ablation studies are presented in the first part in Table 3 (row (a-b)). We observe that the performance on 4 out of 5 datasets suffers a larger decrease by discarding sequential learning (row (a)) than discarding graph-based learning (row (b)). We attribute this to two reasons: (i) The pretrained TransE embeddings have already encoded certain graph-based knowledge into the node representations. (ii) Since there are relatively larger number of attributes in these four datasets than LastFM, it is more important to model the interrelationships among diverse user-accepted attributes for capturing user preferences.

5.4.2 ***Pre-training Embeddings.*** The second part in Table 3 (row (c-d)) presents the results that we replace pretrained TransE embeddings with randomly initialized embeddings (row (c)) and pretrained embeddings from FM model described in Lei et al. [12] (row (d)). We observe a substantial performance gap between UNICORN with randomly initialized embeddings and that with pretrained embeddings, either TransE or FM. This result shows that



(a) UNICORN (b) SCPR (c) EAR (d) CRM

Figure 6: Sample conversations generated by UNICORN, SCPR, EAR, and CRM. Due to the space limit, we only show the conversation at the first five turns. $|V_{\text{cand}}|$ denotes the number of candidate items at the current turn.

Table 4: The effect of hyper-parameters.

Dataset	LastFM*			E-Commerce		
K_v	10	20	50	10	20	50
K_p=1	0.333	0.321	0.306	0.202	0.194	0.186
K_p=10	**0.349**	0.305	0.297	**0.217**	0.205	0.189
K_p=20	0.298	0.286	0.254	0.202	0.199	0.178
L_g	1	2	3	1	2	3
L_s=1	0.320	**0.349**	0.341	0.202	0.217	0.212
L_s=2	0.324	0.346	0.332	0.182	**0.221**	0.216
L_s=3	0.315	0.340	0.339	0.189	0.208	0.196

pretraining is necessary and simply online training with interactive conversations is far from efficient and sufficient for training a CRS with limited interaction data. In addition, using TransE embeddings further improves the performance from FM embeddings to a great extent, which also validates the advantages of graph-enhanced RL.

5.4.3 ***Action Selection Strategies.*** The last part in Table 3 (row (e-h)) presents the results that we replace or discard the proposed action selection strategies. One alternative attribute selection strategy is to adopt the same strategy as the preference-based item selection by changing the object from items to attributes. Another one is to use the original maximum entropy function [12]. The results (row (e,f)) show that the performance of UCRPL suffers a noticeable decrease when adopting the preference-based or entropy-based strategy, indicating that it is required to consider both the user preference and the capability of reducing candidate uncertainty when deciding the asked attribute. Without attribute selection, we observe that the impact on applications with small action space (e.g., LastFM and Yelp) is less than those with large action space (e.g., LastFM*, Yelp*, and E-Commerce). UNICORN is merely working without item selection, since there are no pretrained recommendation components in the framework and the preference-based item selection serves as an auxiliary item recall process.

5.5 Parameter Sensitivity Analysis

The upper part in Table 4 summarizes the experimental results (hDCG) by varying the number of selected candidate actions. Since similar conclusions can also be drawn on other datasets, we only report the results on LastFM* and E-Commerce datasets due to space limit. As for the number of selected attributes, it is likely to discard the important attributes when only selected the attribute with the highest weighted entropy (e.g., K_p=1). However, within a certain training interaction period (10,000 episodes in our case), UNICORN generally achieves the best performance when only selecting a small number of candidate items for policy learning. The results also demonstrate the necessity of pruning the available actions when there is a large action search space in UCRPL.

The lower part shows the experimental results by varying the number of network layers. As for GCN, 1-hop aggregation (L_g=1) is not enough for capturing all the useful information for graph representational learning. However, there is no much difference between 2-hop (L_g=2) and 3-hop (L_g=3). Besides, for Transformer, we can see that 1-layer (L_s=1) is sufficient for a good performance.

5.6 Qualitative Analysis

In order to intuitively study the difference between the proposed UNICORN and other state-of-the-art CRS methods, we randomly sample a real-world interaction from the E-Commerce dataset. The generated conversations by UNICORN, SCPR, EAR, and CRM with the user simulator are presented in Figure 6. Facing the large candidate action space, CRM tends to only trigger the recommendation component to make recommendations, and EAR continuously asks the questions that are not preferred by the user. Despite the success of SCPR in predicting user-preferred attributes, the policy learning in SCPR only decides when to ask or recommend, based on the number of candidate items, which leads to some unnecessary or redundant question-asking turns. UNICORN systematically addresses these issues by making a comprehensive decision of the next action.

6 CONCLUSIONS

In this work, we formulate three separated decision-making processes in CRS, including when to ask or recommend, what to ask and which to recommend, as a unified policy learning problem. To tackle the unified conversational recommendation policy learning problem, we propose a novel and adaptive RL framework, which is based on a dynamic weighted graph. In addition, we further design two simple yet effective action selection strategies to handle the sample efficiency issue. Experimental results show that the proposed method significantly outperforms state-of-the-art CRS methods across four benchmark datasets and the real-world E-Commerce application with remarkable scalability and stability.

REFERENCES

[1] Richard Bellman and Robert Kalaba. 1957. On the role of dynamic programming in statistical communication theory. *IRE Trans. Inf. Theory* 3, 3 (1957), 197–203.

[2] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013*. 2787–2795.

[3] Konstantina Christakopoulou, Alex Beutel, Rui Li, Sagar Jain, and Ed H. Chi. 2018. Q&R: A Two-Stage Approach toward Interactive Recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018*. 139–148.

[4] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. 2016. Towards Conversational Recommender Systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 815–824.

[5] Chongming Gao, Wenqiang Lei, Xiangnan He, Maarten de Rijke, and Tat-Seng Chua. 2021. Advances and Challenges in Conversational Recommender Systems: A Survey. *CoRR* abs/2101.09459 (2021).

[6] Xu Han, Shulin Cao, Xin Lv, Yankai Lin, Zhiyuan Liu, Maosong Sun, and Juanzi Li. 2018. OpenKE: An Open Toolkit for Knowledge Embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations*. 139–144.

[7] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020*. 639–648.

[8] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017*. 173–182.

[9] 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, SIGIR 2018*. 505–514.

[10] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017*.

[11] Wenqiang Lei, Xiangnan He, Maarten de Rijke, and Tat-Seng Chua. 2020. Conversational Recommendation: Formulation, Methods, and Evaluation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020*. 2425–2428.

[12] Wenqiang Lei, Xiangnan He, Yisong Miao, Qingyun Wu, Richang Hong, Min-Yen Kan, and Tat-Seng Chua. 2020. Estimation-Action-Reflection: Towards Deep Interaction Between Conversational and Recommender Systems. In *WSDM '20: The Thirteenth ACM International Conference on Web Search and Data Mining*. 304–312.

[13] Wenqiang Lei, Xisen Jin, Min-Yen Kan, Zhaochun Ren, Xiangnan He, and Dawei Yin. 2018. Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1437–1447.

[14] Wenqiang Lei, Gangyi Zhang, Xiangnan He, Yisong Miao, Xiang Wang, Liang Chen, and Tat-Seng Chua. 2020. Interactive Path Reasoning on Graph for Conversational Recommendation. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2073–2083.

[15] Yu Lei, Hongbin Pei, Hanqi Yan, and Wenjie Li. 2020. Reinforcement Learning based Recommendation with Graph Convolutional Q-network. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020*. 1757–1760.

[16] Raymond Li, Samira Ebrahimi Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. 2018. Towards Deep Conversational Recommendations. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*. 9748–9758.

[17] Shijun Li, Wenqiang Lei, Qingyun Wu, Xiangnan He, Peng Jiang, and Tat-Seng Chua. 2020. Seamlessly Unifying Attributes and Items: Conversational Recommendation for Cold-Start Users. arXiv:cs.IR/2005.12979

[18] Zeming Liu, Haifeng Wang, Zheng-Yu Niu, Hua Wu, Wanxiang Che, and Ting Liu. 2020. Towards Conversational Recommendation over Multi-Type Dialogs. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020*. 1036–1049.

[19] Wenchang Ma, Ryuichi Takanobu, Minghao Tu, and Minlie Huang. 2020. Bridging the Gap between Conversational Reasoning and Interactive Recommendation. *CoRR* abs/2010.10333 (2020).

[20] 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 *The World Wide Web Conference, WWW 2019*. 1210–1221.

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nat.* 518, 7540 (2015), 529–533.

[22] Changhua Pei, Xinru Yang, Qing Cui, Xiao Lin, Fei Sun, Peng Jiang, Wenwu Ou, and Yongfeng Zhang. 2019. Value-aware Recommendation based on Reinforcement Profit Maximization. In *The World Wide Web Conference, WWW 2019*. 3123–3129.

[23] Steffen Rendle. 2010. Factorization Machines. In *ICDM 2010, The 10th IEEE International Conference on Data Mining*. 995–1000.

[24] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2016. Prioritized Experience Replay. In *4th International Conference on Learning Representations, ICLR 2016*.

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

[26] Yueming Sun and Yi Zhang. 2018. Conversational Recommender System. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018*. 235–244.

[27] Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Advances in Neural Information Processing Systems 12*. 1057–1063.

[28] Hado van Hasselt, Arthur Guez, and David Silver. 2016. Deep Reinforcement Learning with Double Q-Learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. 2094–2100.

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*. 5998–6008.

[30] Pengfei Wang, Yu Fan, Long Xia, Wayne Xin Zhao, ShaoZhang Niu, and Jimmy Huang. 2020. KERL: A Knowledge-Guided Reinforcement Learning Model for Sequential Recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020*. 209–218.

[31] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019*. 165–174.

[32] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable Reasoning over Knowledge Graphs for Recommendation. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*. 5329–5336.

[33] Xiang Wang, Yaokun Xu, Xiangnan He, Yixin Cao, Meng Wang, and Tat-Seng Chua. 2020. Reinforced Negative Sampling over Knowledge Graph for Recommendation. In *WWW '20: The Web Conference 2020*. 99–109.

[34] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. 2016. Dueling Network Architectures for Deep Reinforcement Learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016*. 1995–2003.

[35] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. 1112–1119.

[36] 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 2019*. 285–294.

[37] Xin Xin, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M. Jose. 2020. Self-Supervised Reinforcement Learning for Recommender Systems. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020*. 931–940.

[38] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative Knowledge Base Embedding for Recommender Systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 353–362.

[39] Ruiyi Zhang, Tong Yu, Yilin Shen, Hongxia Jin, and Changyou Chen. 2019. Text-Based Interactive Recommendation via Constraint-Augmented Reinforcement Learning. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*. 15188–15198.

[40] Xiaoying Zhang, Hong Xie, Hang Li, and John C. S. Lui. 2020. Conversational Contextual Bandit: Algorithm and Application. In *WWW '20: The Web Conference 2020*. 662–672.

[41] Yongfeng Zhang, Xu Chen, Qingyao Ai, Liu Yang, and W. Bruce Croft. 2018. Towards Conversational Search and Recommendation: System Ask, User Respond. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018*. 177–186.

[42] 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 2020*. 239–248.

[43] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Kaiyuan Li, Yushuo Chen, Yujie Lu, Hui Wang, Changxin Tian, Xingyu Pan, Yingqian Min, Zhichao Feng, Xinyan Fan, Xu Chen, Pengfei Wang, Wendi Ji, Yaliang Li, Xiaoling Wang, and Ji-Rong Wen. 2020. RecBole: Towards a Unified, Comprehensive and Efficient Framework for Recommendation Algorithms. *CoRR* abs/2011.01731 (2020).

[44] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. 2018. Recommendations with Negative Feedback via Pairwise Deep Reinforcement Learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018*. 1040–1048.

[45] 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, WWW 2018*. 167–176.

[46] Lei Zheng, Chun-Ta Lu, Fei Jiang, Jiawei Zhang, and Philip S. Yu. 2018. Spectral collaborative filtering. In *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018*. 311–319.

[47] Kun Zhou, Wayne Xin Zhao, Shuqing Bian, Yuanhang Zhou, Ji-Rong Wen, and Jingsong Yu. 2020. Improving Conversational Recommender Systems via Knowledge Graph based Semantic Fusion. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1006–1014.

[48] Kun Zhou, Yuanhang Zhou, Wayne Xin Zhao, Xiaoke Wang, and Ji-Rong Wen. 2020. Towards Topic-Guided Conversational Recommender System. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, 2020*. 4128–4139.

[49] Sijin Zhou, Xinyi Dai, Haokun Chen, Weinan Zhang, Kan Ren, Ruiming Tang, Xiuqiang He, and Yong Yu. 2020. Interactive Recommender System via Knowledge Graph-enhanced Reinforcement Learning. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020*. 179–188.

[50] Jie Zou, Yifan Chen, and Evangelos Kanoulas. 2020. Towards Question-based Recommender Systems. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020*. 881–890.

[51] Lixin Zou, Long Xia, Pan Du, Zhuo Zhang, Ting Bai, Weidong Liu, Jian-Yun Nie, and Dawei Yin. 2020. Pseudo Dyna-Q: A Reinforcement Learning Framework for Interactive Recommendation. In *WSDM '20: The Thirteenth ACM International Conference on Web Search and Data Mining*. 816–824.