11-2023

Leveraging long short-term user preference in conversational recommendation via multi-agent reinforcement learning

Yang DENG
Singapore Management University, ydeng@smu.edu.sg

Yaliang LI

Bolin DING

Wai LAM

Citation

DENG, Yang; LI, Yaliang; DING, Bolin; and LAM, Wai. Leveraging long short-term user preference in conversational recommendation via multi-agent reinforcement learning. (2023). *IEEE Transactions on Knowledge and Data Engineering*. 35, (11), 11541-11555.
Available at: https://ink.library.smu.edu.sg/sis_research/9088

Leveraging Long Short-Term User Preference in Conversational Recommendation via Multi-agent Reinforcement Learning

Yang Deng, Yaliang Li, Bolin Ding, and Wai Lam, *Senior Member, IEEE*

Abstract—Conversational recommender systems (CRS) endow traditional recommender systems with the capability of dynamically obtaining users' short-term preferences for items and attributes through interactive dialogues. There are three core challenges for CRS, including the intelligent decisions for what attributes to ask, which items to recommend, and when to ask or recommend, at each conversation turn. Previous methods mainly leverage reinforcement learning (RL) to learn conversational recommendation policies for solving one or two of these three decision-making problems in CRS with separated conversation and recommendation components. These approaches restrict the scalability and generality of CRS and fall short of preserving a stable training procedure. In the light of these challenges, we tackle these three decision-making problems in CRS as a unified policy learning task. In order to leverage different features that are important to each sub-problem and facilitate better unified policy learning in CRS, we propose two novel multi-agent RL-based frameworks, namely Independent and Hierarchical Multi-Agent UNIfied COnversational RecommeNders (IMA-UNICORN and HMA-UNICORN), respectively. In specific, two low-level agents enrich the state representations for attribute prediction and item recommendation, by combining the long-term user preference information from the historical interaction data and the short-term user preference information from the conversation history. A high-level meta agent is responsible for coordinating the low-level agents to adaptively make the final decision. Experimental results on four benchmark CRS datasets and a real-world E-Commerce application show that the proposed frameworks significantly outperform state-of-the-art methods. Extensive analyses further demonstrate the superior scalability of the MARL frameworks on the multi-round conversational recommendation.

Index Terms—Conversational recommender system, multi-agent reinforcement learning, graph representation learning

1 Introduction

Conversational Recommender Systems (CRS), which aim to make recommendations by learning user's preferences through multi-turn dialogues [1], [2], [3], have become one of the trending research topics for recommender systems and are gaining increasing attention. Traditional recommender systems [4], [5], [6] or interactive recommender systems (IRS) [7], [8] mainly focus on solving the problem of (i) which items to recommend. There exists generally the other two core research questions for CRS [9], namely (ii) what questions to ask and (iii) when to ask or recommend. The CRS has the natural advantage of explicitly acquiring user's preferences and revealing the reasons behind the recommendation. Recent works have demonstrated the importance of asking clarifying questions in CRS [1], [10], [11]. More importantly, deciding when to ask or recommend is the key to coordinating conversation and recommendation for developing an effective CRS [2], [12], [13]. Different problem settings of CRS have been proposed, either from the perspective of dialog systems, being a variation of task-oriented dialog [3], [14], or from the perspective of recommender systems, being an enhanced interactive recommender system [2], [12], [15]. In this work, we study the multi-round conversational recommendation (MCR) setting [2], where the system asks questions about users' preferences on certain attributes or recommends items multiple times, with the goal of making a 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 [2], [12], [13]. Early studies on MCR typically target at solving one or two core research questions. For example, CRM [12] and EAR [2] employ policy gradient [16] 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, SCPR [13] and SeqCR [17] only consider 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. Despite the effectiveness of these methods, there are some challenges that remained to be tackled for real-world applications: (i) Models trained with existing CRS methods lack generality to different domains or applications, since there are three separated decision-

- *Yang Deng and Wai Lam are with the Department of System Engineering and Engineering Management, The Chinese University of Hong Kong, Hong Kong 999077. E-mail: {ydeng, wlam}@se.cuhk.edu.hk.*
- *Yaliang Li and Bolin Ding are with Alibaba Group, Bellevue, WA 98004 USA. E-mail: {yaliang.li, bolin.ding}@alibaba-inc.com.*

Manuscript received 13 December 2021; revised 14 November 2022; accepted 20 November 2022. Date of publication 28 November 2022; date of current version 6 October 2023.
This work was supported by the Research Grant Council of the Hong Kong Special Administrative Region, China under Grant 14200719.
(Corresponding author: Yang Deng.)
Recommended for acceptance by E. Chen.
Digital Object Identifier no. 10.1109/TKDE.2022.3225109

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 [2], [13] or pretrain the policy network with synthetic dialogue history [2], [12]. (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 this end, our recent work [18] formulated the aforementioned three separated decision-making processes in CRS as a unified policy learning problem. This can not only harness the ultimate goal of CRS but also 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 asking an attribute or recommending items, at each conversation turn to maximize the cumulative utility over the whole MCR process. For the UCRPL problem, we proposed a novel and adaptive graph-based reinforcement learning framework, namely UNICORN. Specifically, we employ the graph structure, capturing the rich correlated information among different types of nodes (i.e., users, items, and attributes), to discover collaborative user preferences towards attributes and items. The recommendation and conversation components are integrated as an organic whole, where the conversation session is regarded as a sequence of nodes maintained in the graph. Then the conversation history can be dynamically exploited for predicting the action at the next turn.

However, during the online training of the UNICORN framework, we encounter two practical issues that can be addressed for further enhancement: (i) Since both attributes and items are equally regarded as possible actions that can be taken by the same RL agent, it is difficult to introduce separated features that might be beneficial to either attribute prediction or item recommendation, respectively. For instance, the historical interactions, e.g., purchase history, have been identified to attach great importance in determining which items to recommend for CRS [17], [19]. This kind of historical interaction data is typically adopted for modeling the long-term user preference in traditional recommender systems [20], [21]. Besides, there often exists user profile data, e.g., user-liked attribute [13], which can also benefit the decision of what questions to ask. (ii) UNICORN only considers the user preference towards each candidate attribute or item at each conversation turn, but neglects some global features that are also valuable and important in determining the timing of recommendation, such as the user feedback history or the number of candidate items. Therefore, UNICORN may continuously recommend items even when encountering successive failures or there are still a large number of candidate items, or keep asking questions when the candidate items have already been narrowed down. Therefore, in order to adaptively incorporate different features that are important to each sub-problem and facilitate better unified conversational recommendation policy learning, we investigate the utilization of multi-agent reinforcement learning (MARL) in CRS.

To this end, in this paper, we substantially extend the UNICORN framework to be a multi-agent RL-based framework, including two variants, namely Independent Multi-Agent (IMA-UNICORN) and Hierarchical Multi-Agent UNIfied COnversational RecommeNders (HMA-UNICORN). In specific, 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. We integrate graph-enhanced representation learning and sequential conversation modeling to capture dynamic short-term user preferences towards items and attributes along with the ongoing conversation as the generic state representation. Then two agents for attribute prediction and item recommendation further combine long-term user preference information from the historical interaction data with the generic state representation into the agent-specific state representations, respectively. A meta-agent is introduced to decide the final action that conditions on the meta state and the actions of all agents. Finally, two types of multi-agent Q-learning frameworks are proposed to facilitate the unified conversational recommendation policy learning by soliciting both long-term and short-term user preference information.

The main contributions are summarized as follows:

- To the best of our knowledge, this is the first attempt to study the utility of multi-agent reinforcement learning for solving CRS problems. We propose two novel MARL frameworks for the unified policy learning problem in multi-round conversational recommendation, including Independent and Hierarchical Multi-Agent Q-learning.
- We propose to enrich the state representation learning by combining long-term user preference information from the historical interaction data and short-term user preference information from the conversation history.
- Experimental results show that the proposed frameworks significantly outperform state-of-the-art CRS methods across four public benchmark datasets and a real-world E-Commerce application.

2 Related Works

2.1 Conversational Recommendation

Based on the problem settings, current CRS studies can be categorized into four directions [9], [22]: (1) Exploration-Exploitation Trade-offs for Cold-start Users [10], [23], [24]. 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 [1], [11], [15] aim at asking questions to users to get more information about their preferences, which is often addressed as "asking clarifying question". (3) Dialogue Understanding and Generation [3], [25], [26]. These studies focus on how to understand users' preferences and intentions from their utterances and generate fluent responses so as to deliver natural dialogue actions. (4) Multi-round Conversational Recommendation [2], [12], [13]. 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 settings, we focus on the MCR problem.

Most of existing CRS studies [2], [13], [18] focus on exploring the real-time user preferences during the on-going conversations for making better recommendations. Motivated by session-based recommendation studies [27], [28], some latest works [14], [17], [19] identify that the long-term user preferences from historical interaction data also attach great importance in making a better decision of which items to recommend for CRS. In this work, we investigate the combination of short-term and long-term user preferences for improving the unified policy learning in CRS.

2.2 RL in Recommendation

Reinforcement learning (RL) has been widely introduced into recommender systems due to its advantage of considering users' long-term feedbacks [29], [30]. 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 [29], [30], [31], [32]. Recent works on sequential recommendation [33], [34] and interactive recommendation [7], [8], [35] 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 [2], [12] and (ii) when to ask or recommend [13]. 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.

2.3 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 [36], [37], [38], and adopting knowledge graph embeddings as rich context information [39], [40]. The other group of studies models recommendation as a path reasoning problem for building explainable recommender systems [41], [42]. Recent years have witnessed many successful applications of graph-based RL methods on different scenarios of recommender systems [8], [13], [43], [44], [45], [46]. For example, [44] 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 [46]. [45] and [8] employ graph convolutional network (GCN) [47] for state representation learning to enhance the performance of traditional RL methods on recommendation policy learning.

2.4 Multi-Agent Reinforcement Learning (MARL)

MARL algorithms [48] are generally categorized into four groups according to fully cooperative, fully competitive, both cooperative and competitive, and neither cooperative nor competitive tasks. The simplest approach to realizing a MARL algorithm is to learn each agent independently [49]. However, the independent agents are not able to coordinate their actions, failing to achieve complicated cooperation. To achieve agents' cooperation, several attempts have been made on learning communication among multiple agents [50] or employing the centralized training decentralized execution architecture [51]. For instance, in multi-module recommendation, the MASSA approach [52] restricts different modules not to communicate with each other and a signal network is developed to promote cooperation by generating signals for different modules. Besides, different centralized critics [53], [54] are designed to coordinate the recommendation between multiple agents. In this work, we make the first attempt to introduce MARL into CRS and investigate the utilization of two kinds of MARL frameworks for the policy learning of CRS.

3 Preliminary

3.1 Problem Definition

Multi-Round Conversational Recommendation. In this work, we focus on the multi-round conversational recommendation (MCR) scenario [2], [13], 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. For the user u, we can obtain an interaction sequence of her/his historical items $\mathcal{V}_u$ and a set of her/his liked attributes $\mathcal{P}_u^*$. 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 [2], 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. MCR 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

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

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

3.2 Graph-Based MDP Environment

The MDP environment is responsible for informing the agent about the current state and possible actions to take,

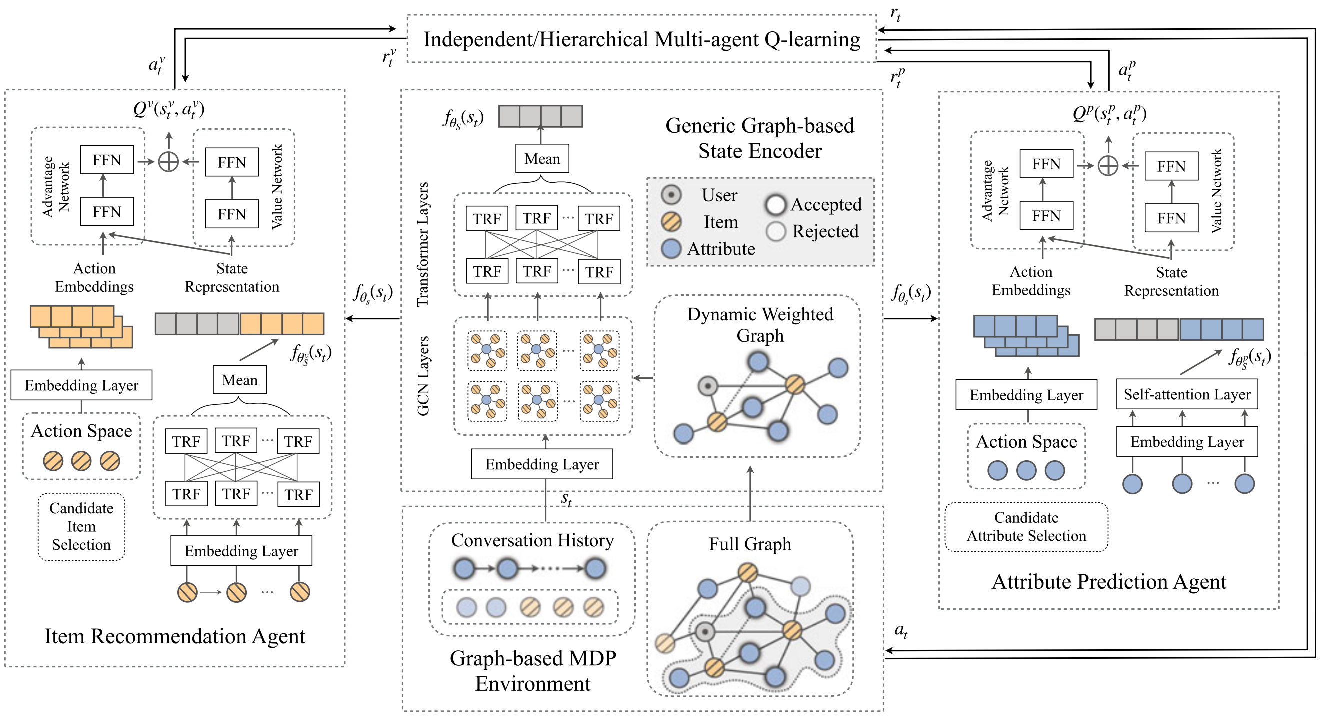

Fig. 1. Overview of the proposed frameworks, namely IMA-UNICORN and HMA-UNICORN.

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, $\mathcal{T}: \mathcal{S}\times\mathcal{A}\rightarrow\mathcal{S}$ refers to the state transition function, and $\mathcal{R}: \mathcal{S}\times\mathcal{A}\rightarrow\mathbb{R}$ is the reward function.

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. $\mathcal{P}_u$ 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., $s_0 = [[\{p_0\},\{\},\{\}], \mathcal{G}_u^{(0)}]$.

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 [13], 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).

*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 $[\mathcal{H}_u^{(t+1)}, \mathcal{G}_u^{(t+1)}]$.

Reward. Following previous MCR studies [2], [13], 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.

3.3 Dynamic Graph Construction

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 the 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

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

where $\sigma(\cdot)$ denotes the sigmoid function, e_u, e_v, and e_p are the embeddings of the user, item, and attribute, respectively.

4 Proposed Frameworks

The overview of the proposed frameworks, IMA-UNICORN and HMA-UNICORN, are depicted in Fig. 1, which consist of four main components:

1) *Generic Graph-based State Encoder* models the dynamic short-term user preference towards items and attributes along with the conversation over a graph-based MDP environment;
2) *Item Recommendation Agent* captures the sequential information of users' historical interactions for long-term preference modeling and performs candidate item selection;
3) *Attribute Prediction Agent* models the static long-term user preferences towards attributes from user profiling data and performs candidate attribute selection;
4) *Multi-agent Q-learning*. We investigate two types of multi-agent Q-learning, i.e., Independent Multi-agent Q-learning and Hierarchical Multi-agent Q-learning.

4.1 Generic Graph-Based State Encoder

We first employ a generic graph-based state encoder to model the dynamic short-term user preference towards items and attributes along with the conversation. 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 [55], [56] to obtain node embeddings for all the nodes in the full graph $\mathcal{G}$.

4.1.1 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) [47] 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:

$$e_i^{(l+1)} = \text{ReLU}\Big(\sum\nolimits_{j\in\mathcal{N}_i} \mathbf{\Lambda}_{i,j}\boldsymbol{W}_l e_j^{(l)} + \boldsymbol{B}_l e_i^{(l)}\Big), \tag{7}$$

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 $\mathbf{\Lambda}$ is a normalization adjacent matrix as $\mathbf{\Lambda} = \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$ with $\boldsymbol{D}_{ii} = \sum_j \boldsymbol{A}_{i,j}$.

4.1.2 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 [2], [13] that adopt heuristic features for conversation history modeling, we employ Transformer encoder [57] for capturing the sequential information of the conversation history as well as attending the important information for deciding the next action. As described in [57], 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 with the learned graph-based representation $\{e_p^{(L_g)} : p \in \mathcal{P}_u^{(t)}\}$ and the learnable position embeddings, where L_g is the number of layers in GCN. After the sequential learning with L_s Transformer layers, we 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 generic 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.

4.2 Item Recommendation Agent

4.2.1 State Encoder With Long-Term Preference

Inspired by sequential recommendation [58], several previous studies [17], [19] identify that the interaction sequence of historical items can improve the performance of recommendation in CRS. Therefore, we employ another Transformer encoder to capture the sequential information of the user's historical interactions for long-term preference modeling:

$$\boldsymbol{E}^* = \text{MultiHead}(\boldsymbol{E}^{(l)},\boldsymbol{E}^{(l)},\boldsymbol{E}^{(l)}), \tag{11}$$

$$\boldsymbol{E}^{(l+1)} = \text{LayerNorm}(\text{FFN}(\boldsymbol{E}^*) + \boldsymbol{E}^{(l)}), \tag{12}$$

where $\boldsymbol{E} \in \mathbb{R}^{|\mathcal{V}_u|\times d}$ denotes the embeddings. Here the input sequence $\boldsymbol{E}^{(0)}$ is the user's historical interacted items with the pre-trained graph embeddings $\{e_v : v \in \mathcal{V}_u\}$ and the learnable position embeddings. After the sequential learning with L_v Transformer layers, we can aggregate the long-short term user preference information learned from both the generic graph-based state encoder and the agent-specific state encoder for the state representation $f_{\theta_S^v}(s_t)$ of the item recommendation agent:

$$f_{\theta_S^v}(s_t) = [f_{\theta_S}(s_t); \mathrm{MeanPool}(\boldsymbol{E}^{L_v})], \tag{13}$$

where [:] denotes the concatenate operation.

4.2.2 Candidate Item Selection

A large action search space will harm the performance of the policy learning to a great extent [13]. Thus, it attaches great importance to handling the overwhelmingly large action space in UCRPL. 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 $\mathcal{A}_t^v$ at each timestep t, which is ranked by the recommendation score $w_v^{(t)}$ in Eq.(6).

Algorithm 1. Independent Multi-Agent Q-Learning for Multi-Round Conversational Recommendation

```
    Input: $\{\mathbf{e}_i\}_{i\in\mathcal{N}}$; $\mathcal{D}_p$; $\mathcal{D}_v$; $\tau$; $\epsilon$; $\gamma$; $K$; $T$; $K_v$; $K_p$;
    Output: $\theta_S^v$; $\theta_S^p$; $\theta_Q^v$; $\theta_Q^p$;
 1: Initialize all parameters: $\theta_S^v$, $\theta_S^p$, $\theta_Q^v$, $\theta_Q^p$, $\theta_Q^v{}' \leftarrow \theta_Q^v$, $\theta_Q^p{}' \leftarrow \theta_Q^p$;
 2: for $episode = 1, 2, \ldots, N$ do
 3:   User $u$ specifies an attribute $p_0$ to start;
 4:   Update: $\mathcal{P}_u^{(0)} = \{p_0\}$,  $\mathcal{P}_{\text{rej}}^{(0)} = \{\}$,  $\mathcal{V}_{\text{rej}}^{(0)} = \{\}$,
      $\mathcal{H}_u^{(0)} = [\mathcal{P}_u^{(0)}, \mathcal{P}_{\text{rej}}^{(0)}, \mathcal{V}_{\text{rej}}^{(0)}]$,   $s_0 = [\mathcal{H}_u^{(0)}, \mathcal{G}_u^{(0)}]$;
 5:   Get candidate action space $\mathcal{A}_0^p$, $\mathcal{A}_0^v$ via Action Selection w.
      r.t. Eq.(15) and Eq.(6);
 6:   for turn $t = 0, 1, \ldots, T-1$ do
 7:     // Transition Generation;
 8:     for each agent $x \in \{p, v\}$ do
 9:       Get state representation $f_{\theta_S^x}(s_t)$ via Eq.(14) or Eq.(13);
10:       Select the action $a_t^x$ by $\epsilon$-greedy w.r.t Eq.(17);
11:       Receive reward $r_t^x$ from environment;
12:       Update the next state $s_{t+1}^x = \mathcal{T}(s_t, a_t^x)$;
13:       Get $\mathcal{A}_{t+1}^x$ via Action Selection;
14:       Store $(s_t, a_t^x, r_t^x, s_{t+1}^x, \mathcal{A}_{t+1}^x)$ to buffer $\mathcal{D}_x$;
15:     end
16:     Select the action $a_t$ w.r.t. Eq.(17);
17:     Receive reward $r_t$ from environment;
18:     Update the next state $s_{t+1} = \mathcal{T}(s_t, a_t)$;
19:     Get $\mathcal{A}_{t+1}$ via Action Selection;
20:     // Parameter Updating;
21:     for each agent $x \in \{p, v\}$ do
22:       Sample mini-batch of $(s_t, a_t, r_t, s_{t+1}, \mathcal{A}_{t+1})$ w.r.t Eq.
          (23);
23:       Compute the target value $y_t$ via Eq. (21);
24:       Update $\theta_S^x$, $\theta_Q^x$ via SGD w.r.t the loss function Eq.(19);
25:       Update $\theta_Q^x{}'$ via Eq.(22) ;
26:     end
27:   end
28: end
```

4.3 Attribute Prediction Agent

4.3.1 State Encoder with Long-term Preference

In practice, those recommendation platforms often maintain a considerable amount of user profiling data, such as user-liked attribute [13]. This kind of data provides prior knowledge about the static long-term user preferences, which can be combined with the dynamic short-term user preferences learned from the current conversation to facilitate better attribute prediction. Thus, we employ a self-attention layer to aggregate the pre-trained graph embeddings of the user-liked attributes $\boldsymbol{E}_{\mathcal{P}_u^*} = \{e_p : p \in \mathcal{P}_u^*\}$. The static and dynamic user preferences are concatenated to form the state representation for the attribute prediction agent:

$$f_{\theta_S^p}(s_t) = [f_{\theta_S}(s_t); \mathrm{MultiHead}(\boldsymbol{E}_{\mathcal{P}_u^*}, \boldsymbol{E}_{\mathcal{P}_u^*}, \boldsymbol{E}_{\mathcal{P}_u^*})]. \tag{14}$$

4.3.2 Candidate Attribute Selection

Similar to candidate item selection, we also impose candidate attribute selection to handle the large action space for attribute prediction. Whereas for candidate attributes to be asked, the expected one is supposed to not only better eliminate the uncertainty of candidate items, but also encode the user preference. Inspired by [13], we adopt weighted entropy as the criteria to prune candidate attributes:

$$w_p^{(t)} = -\mathrm{prob}(p^{(t)}) \cdot \log\left(\mathrm{prob}(p^{(t)})\right), \tag{15}$$

$$\mathrm{prob}(p^{(t)}) = \sum\nolimits_{v\in\mathcal{V}_{\text{cand}}^{(t)}\cap\mathcal{V}_p^{(t)}} w_v^{(t)} \Big/ \sum\nolimits_{v\in\mathcal{V}_{\text{cand}}^{(t)}} w_v^{(t)}, \tag{16}$$

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

4.4 Independent Multi-Agent Q-Learning

One natural approach for the multi-agent RL problem is independent Q-learning [49], which assumes that each agent, including the agent for attribute prediction and the agent for item recommendation, ignores the existence of the other agent in the environment and learns its own Q-function that conditions on the state and its own action without communication. After obtaining the state representation and the candidate action space, we introduce the deep Q-learning network (DQN) [59] to conduct the policy learning for each single agent. In order to enhance and stabilize the training of DQN, we further incorporate some RL techniques into the implementation.

4.4.1 Single-Agent Deep Q-Learning Network

In the MCR scenario, two agents discussed above share the same DQN structure and loss function, but are updated independently. Here we adopt the same notations to introduce the single-agent DQN for simplicity.

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 Fig. 1, the dueling Q-network [60] 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:

$$Q(s_t, a_t) = f_{\theta_V}(a_t) + f_{\theta_A}(f_{\theta_S}(s_t), a_t), \tag{17}$$

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 $\theta_Q = \{\theta_V, \theta_A\}$.

Algorithm 2. Hierarchical Multi-Agent Q-Learning for Multi-Round Conversational Recommendation

```
Input: ${\mathbf{e}_i}_{i\in\mathcal{N}}$; $\mathcal{D}_h$; $\mathcal{D}_p$; $\mathcal{D}_v$; $\tau$; $\epsilon$; $\gamma$; $K$; $T$; $K_v$; $K_p$;
Output: $\theta_S^p$; $\theta_S^v$; $\theta_Q^h$; $\theta_Q^p$; $\theta_Q^v$;
1: Initialize all parameters: $\theta_S^p$, $\theta_S^v$, $\theta_Q^h$, $\theta_Q^p$, $\theta_Q^v$, ${\theta_Q^h}' \leftarrow \theta_Q^h$, ${\theta_Q^p}' \leftarrow \theta_Q^p$, ${\theta_Q^v}' \leftarrow \theta_Q^v$;
2: for $episode = 1, 2, \ldots, N$ do
3:   User $u$ specifies an attribute $p_0$ to start;
4:   Update: $\mathcal{P}_u^{(0)} = \{p_0\}$, $\mathcal{P}_{\text{rej}}^{(0)} = \{\}$, $\mathcal{V}_{\text{rej}}^{(0)} = \{\}$, $\mathcal{H}_u^{(0)} = [\mathcal{P}_u^{(0)}, \mathcal{P}_{\text{rej}}^{(0)}, \mathcal{V}_{\text{rej}}^{(0)}]$, $s_0^l = [\mathcal{H}_u^{(0)}, \mathcal{G}_u^{(0)}]$;
5:   Get candidate action space $\mathcal{A}_0^p$, $\mathcal{A}_0^v$ via Action Selection w.r.t. Eq.(15) and Eq.(6);
6:   for $turn\ t = 0, 1, \ldots, T-1$ do
7:     // Transition Generation;
8:     Get state representation $f_{\theta_S^p}(s_t^l)$ via Eq.(14);
9:       Compute the Q-value $Q^p(s_t^l, a_t)$ for each candidate action $a_t^p \in \mathcal{A}_t^p$;
10:    Get state representation $f_{\theta_S^v}(s_t^l)$ via Eq.(13);
11:      Compute the Q-value $Q^v(s_t^l, a_t)$ for each candidate action $a_t^v \in \mathcal{A}_t^v$;
12:    Get meta state $s_t^h$ via Eq.(24);
13:    Select the meta action $a_t^h$ by $\epsilon$-greedy w.r.t $\arg\max_{a_t\in\mathcal{A}_t^h} Q^h(s_t^h, a_t)$;
14:    if $a_t^h == a_{ask}$ then
15:      $a_t^l = \arg\max_{a_t\in\mathcal{A}_t^p} Q^p(s_t^l, a_t)$
16:    else
17:      $a_t^l = \arg\max_{a_t\in\mathcal{A}_t^v} Q^v(s_t^l, a_t)$;
18:    else
19:      Receive reward $r_t$ from environment, and intrinsic rewards $r_t^v$ or $r_t^p$ via Eq.(25);
20:    Update the next state $s_{t+1}^l = \mathcal{T}(s_t^l, a_t^l)$;
21:    Update the next meta state $s_{t+1}^h$;
22:    Get $\mathcal{A}_{t+1}^p$, $\mathcal{A}_{t+1}^v$ via Action Selection;
23:    Store $(s_t^h, a_t^h, r_t, s_{t+1}^h)$ to buffer $\mathcal{D}_h$;
24:    if $a_t^h == a_{ask}$ then
25:      Store $(s_t^l, a_t^l, r_t, s_{t+1}^l, \mathcal{A}_{t+1}^p)$ to buffer $\mathcal{D}_p$;
26:    else
27:      Store $(s_t^l, a_t^l, r_t, s_{t+1}^l, \mathcal{A}_{t+1}^v)$ to buffer $\mathcal{D}_v$;
28:    end
29:    // Parameter Updating;
30:    for each agent $x \in \{p, v, h\}$ do
31:      Sample a mini-batch from $\mathcal{D}_x$ w.r.t Eq.(23);
32:      Compute the target value $y_t$ via Eq. (21);
33:      Update $\theta_S^x$, $\theta_Q^x$ via SGD w.r.t the loss function Eq.(19);
34:      Update ${\theta_Q^x}'$ via Eq.(22) ;
35:    end
36:  end
37: end
```

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 [61] as:

$$Q^*(s_t, a_t) = \mathbb{E}_{s_{t+1}}\Big[r_t + \gamma \max_{a_{t+1}\in\mathcal{A}_{t+1}} Q^*(s_{t+1}, a_{t+1}) | s_t, a_t\Big]. \quad (18)$$

During each episode in the MCR process, at each timestep t, the agent obtains the current state representation $f_{\theta_S}(s_t)$ via the corresponding state encoder described in Section 4.2.1 or Section 4.3.1. 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 4.2.2 or Section 4.3.2. 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:

$$\mathcal{L}(\theta_Q, \theta_S) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}, \mathcal{A}_{t+1})\sim\mathcal{D}}\big[(y_t - Q(s_t, a_t; \theta_Q, \theta_S))^2\big], \quad (19)$$

$$y_t = r_t + \gamma \max_{a_{t+1}\in\mathcal{A}_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_Q, \theta_S), \quad (20)$$

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

To alleviate the overestimation bias problem in conventional DQN, we adopt Double Q-learning [62], 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:

$$y_t = r_t + \gamma Q'\big(s_{t+1}, \arg\max_{a_{t+1}\in\mathcal{A}_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_Q, \theta_S); \theta_{Q'}, \theta_S\big), \quad (21)$$

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

$$\theta_{Q'} = \tau\theta_Q + (1-\tau)\theta_{Q'}, \quad (22)$$

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 [63] as a proxy for learning potential, which samples transitions with probability δ relative to the absolute TD error:

$$\delta \propto \Big| r_{t+1} + \gamma Q'\big(s_{t+1}, \arg\max_{a_{t+1}\in\mathcal{A}_{t+1}} Q(s_{t+1}, a_{t+1})\big) - Q(s_t, a_t)\Big|. \quad (23)$$

4.4.2 Independent Multi-Agent Training and Inference

The training procedure of the Independent Multi-agent Q-learning for Multi-round Conversational Recommendation (IMA-UNICORN) is presented in Algorithm 1. At each conversation turn, the item recommendation agent and the attribute prediction agent first interact with the same environment independently to collect their own transitions of episodic experiences. Then we select the real action a_t with the highest Q-value among two agents to proceed the conversation. After all, each agent is trained over the sampled mini-batches of experiences to update the network parameters of DQN for approximating their own Q-function by using the overall loss function in Eq.(19). Although each agent is responsible for maintaining its own network parameters, they share the same graph-based MDP environment and are trained end-to-end as a unified model.

With the learned IMA-UNICORN model, given a user and her/his conversation history, we follow the same process to obtain the candidate action space and the current state representation for each agent, and then decide the next action according to the max Q-value in Eq.(17) among two agents. 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.

4.5 Hierarchical Multi-Agent Q-Learning

Another approach that can be applied to the multi-agent RL problem is to introduce a "meta-agent" that decides the final action that conditions on the meta state and the actions of all agents. Therefore, we further propose a Hierarchical Multi-agent Q-learning method for unified policy learning.

4.5.1 Meta State

Intuitively, the role of the meta-agent is to decide whether to ask questions or recommend items, which is similar to the policy learning problem in [13]. Similarly, the meta-state vector s_t^h first consists of two vectors s_{his} and s_{len}. s_{his} encodes the user's feedback history, which can guide the meta-agent to adjust its strategy, e.g., if the user accepts the asked attributes for multiple turns, it is likely to be a suitable timing to make recommendations. s_{len} encodes the size of the candidate item set. As discussed in [2], it is easier to make successful recommendations when there are fewer candidate items.

Besides these two vectors, the meta-state s_t^h is also supposed to involve the actions selected by the low-level agents, i.e., a_t^p and a_t^v, since it would be better to take the low-level action with higher estimated rewards. To achieve this, we further concatenate the Q-values of the selected actions by each low-level agent into the meta-state vectors:

$$s_t^h = [s_{his}; s_{len}; Q^p(s_t, a_t^p); Q^v(s_t, a_t^v)]. \tag{24}$$

4.5.2 Internal Critic

The internal critic is responsible for providing intrinsic reward r_t^x to a specific low-level agent $x \in \{v, p\}$ after an action a_t^l is taken at turn t. Here we just employ a simple reward function to enable the internal critic as follows:

$$r_t^x = \begin{cases} 0.1, & \text{if } r_t > 0, \quad a_t^h = x, \\ -0.1, & \text{if } r_t < 0, \quad a_t^h = x, \\ 0, & \text{otherwise.} \end{cases} \tag{25}$$

4.5.3 Deep Q-Learning Network for Meta-Agent

The DQN for meta-agent takes the meta-state vector s_t^h as input and outputs the Q-values $Q^h(s_t^h, a_t^h)$ for the two actions, i.e., a_{ask} and a_{rec}, to estimate the reward. In other words, the action space for meta-agent is fixed: $\mathcal{A}_t^h = \{a_{ask}, a_{rec}\}$. We use a two-layer feed forward neural network to approximate the Q-function. Except that there is no candidate action selection, the loss function and other RL techniques are the same as the low-level agent as described in Section 4.4.1.

4.5.4 Hierarchical Multi-Agent Training and Inference

The training procedure of the Hierarchical Multi-agent Q-learning for Multi-round Conversational Recommendation (HMA-UNICORN) is presented in Algorithm 2. At each conversation turn, each low-level agent encodes its state representation and computes the Q-value for each candidate action from its candidate action space. Then, based on the meta state and the selected actions by low-level agents, the high-level meta-agent decides to take the action selected by which low-level agent. After the meta-agent interacts with the environment, the state for each low-level agent as well as the meta state will transit to the next state. The transition of the meta state and the transition of the state of the activated low-level agent will be stored into corresponding buffers. The parameter updating, training and inference processes are the same as IMA-UNICORN as described in Section 4.4.2.

4.6 Complexity Analysis

In this section, we discuss the time complexity of the proposed methods. As shown in Algorithm 1 and 2, the time complexity of IMA-UNICORN and HMA-UNICORN is mainly dependent of two parts: the state representation learning and the computation of Q-value.

For IMA-UNICORN, the state representation learning is in threefold: 1) The time complexity for the graph-based representation learning in Eq.(7) is $\mathcal{O}(L_g|\mathcal{N}|d^2)$, where $|\mathcal{N}|$ is the number of nodes. 2) The time complexity for the sequential representation learning in Eq.(8) is $\mathcal{O}(L_sT^2d)$, where T is the maximum number of conversation turns. 3) The time complexity of long-term preference modeling in item recommendation agent (Eq.(13)) and attribute prediction agent (Eq.(14)) is $\mathcal{O}(L_v|\mathcal{V}_u|^2d)$ and $\mathcal{O}(|\mathcal{P}_u^*|^2d)$, respectively. Besides, the time complexity of the computation of Q-value in Eq. (17) is $\mathcal{O}(d^2)$. To sum up, the time complexity of IMA-UNICORN includes the state representation learning and the computation of Q-value for one of the low-level agent depending on the selected action, i.e., $\mathcal{O}(L_g|\mathcal{N}|d^2 + L_sT^2d + L_v|\mathcal{V}_u|^2d + d^2)$ or $\mathcal{O}(L_g|\mathcal{N}|d^2 + L_sT^2d + |\mathcal{P}_u^*|^2d + d^2)$.

For HMA-UNICORN, the only difference is the incorporation of the meta agent. The time complexity of the additional meta-state representation learning in Eq.(24) is $\mathcal{O}(L_v|\mathcal{V}_u|^2d + |\mathcal{P}_u^*|^2d)$. Besides, it requires to compute three different Q-values for each agent. Therefore, the time complexity of HMA-UNICORN can be estimated as $\mathcal{O}(L_g|\mathcal{N}|d^2 + L_sT^2d + L_v|\mathcal{V}_u|^2d + |\mathcal{P}_u^*|^2d + d^2)$.

Overall, as T, $|\mathcal{V}_u|$, and $|\mathcal{P}_u^*|$ are typically less than $|\mathcal{N}|$, IMA-UNICORN and HMA-UNICORN share a similar time complexity, which mainly depends on the graph-based representation learning module. Therefore, it would be beneficial to leverage a more efficient graph learning algorithm for improving the overall time complexity of the proposed method.

5 Experimental Settings

5.1 Datasets

We evaluate the proposed methods on four existing multi-round conversational recommendation benchmark datasets and a real-world E-Commerce dataset. The statistics of these datasets are presented in Table 1.

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

- *LastFM* and *Yelp*. The LastFM dataset is used for evaluation on music artist recommendation, while the Yelp dataset is for business recommendation. [2] 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**. [13] consider that it is not realistic to manually merge attributes for applications, so they adopt original attributes to reconstruct these two datasets.
- *E-Commerce*. A real-world E-Commerce dataset [18] collected from Taobao.

5.2 Evaluation Metrics

Following previous studies on multi-round conversational recommendation [2], [13], we adopt success rate at the turn t (SR@t) [12] 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. In addition, we also adopt *hNDCG@(T, K)* [18] to conduct a comprehensive evaluation of CRS, which extends the normalized discounted cumulative gain (NDCG@K) to be a two-level hierarchical version. 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 successful turn.

5.3 User Simulator

Due to the interactive nature of MCR, it needs to be trained and evaluated by interacting with users. We adopt the standard user simulator in MCR setting as previous studies [2], [13], [17] to 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" [1] as described in Section 3.1.

5.4 Baselines

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

- *Max Entropy* is a rule-based strategy to choose attributes to ask based on the maximum entropy within the current state or recommend items with a certain probability [2].
- *Abs Greedy* [10] only performs recommendation actions until the CRS makes a successful recommendation or exceeds the maximum turns of conversation.
- *CRM* [12] employs policy gradient [16] to learn the policy deciding when and which attributes to ask.
- *EAR* [2] is a three-stage method to enhance the interaction between the conversation and recommendation components with a similar RL framework as CRM.
- *SCPR* [13] leverages path reasoning on the graph to prune off candidate attributes and adopts the DQN [59] framework to determine when to ask or recommend.
- *SeqCR* [17] incorporates users' historical interactions into the recommendation model in SCPR method to facilitate better recommendation performance.
- *UNICORN* [18] is a graph-based DQN framework to handle the unified conversational recommendation policy learning problem.

5.5 Implementation Details

Following previous studies [2], [13], [18], we set the size K of the recommendation list as 10, the maximum turn T as 15. Different from existing methods for MCR [2], [12], [13], which requires to train an offline recommendation models (e.g., FM) as well as pretrain the policy network with offline dialogue history, we can simply initialize the whole framework with pretrained graph-based embeddings. We adopt TransE [55] from OpenKE [64] to pretrain the node embeddings in the constructed graph with the training set. Following previous studies [2], [13], 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 GCN layers (i.e., L_g) and Transformer layers (i.e., L_s, L_v) 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 0.999 and 0.01.

We adopt the user simulator described in Section 5.3 to interact with the CRS for online training the model using the validation set. Due to the difficulty of the convergence for training RL-based methods, we conduct the online training for the same number of training episodes, i.e., 10,000 episodes, for all implemented methods.

6 Experimental Results and Analyses

6.1 Overall Performance

Table 2 shows the performance comparison between the proposed methods, IMA-UNICORN and HMA-UNICORN, and all baselines across five datasets. Among the baselines, UNICORN not only enables the conversation and recommendation to be mutually enhanced during the training

TABLE 2
Experimental Results

	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 [12]	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 [2]	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 [13]	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
SeqCR [17]	0.501	12.58	0.152	0.739	8.11	0.346	0.976	5.46	0.391	0.478	12.21	0.181	0.545	11.56	0.186
UNICORN [18]	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
IMA-UNICORN	0.542*	11.64*	0.174*	0.777*	7.05*†	0.412*†	0.978*	5.29*	0.399*	0.506*	11.56*	0.190*	0.644*†	11.08*	0.235*†
HMA-UNICORN	**0.576***†	**11.35***†	**0.185***†	**0.824***†	**6.78***†	**0.421***†	**0.986***	**5.02***†	**0.402***†	0.512*	**11.14***	**0.205***	**0.672***†	**10.40***†	**0.249***†

* *indicates statistically significant improvement ($p < 0.05$) over SCPR.* † *indicates statistically significant improvement ($p < 0.05$) over UNICORN. hDCG stands for hDCG@(15,10). SR and hDCG are the higher the better, while AT is the lower the better.*

process, but also attains an effective sample efficiency with the proposed action selection strategies. This leads to a substantial margin from other baselines, about 18% for SR@15, 2 turns for AT, and 30% for hDCG. As for the proposed multi-agent RL-based methods, IMA-UNICORN achieves a competitive performance as UNICORN and effectively shortens the average turn (AT) of making a successful recommendation. Furthermore, HMA-UNICORN improves the performances across all the datasets (except for YELP*) by achieving a significantly higher success rate and less average turn than UNICORN, which is also comprehensively validated by the improvements on hDCG. Since there is no long-term user preference data in YELP dataset, IMA-UNICORN and HMA-UNICORN may not fully make use of the MARL framework to improve the performance of UNICORN on YELP and YELP*.

6.2 Training Efficiency

Fig. 2 shows the test performance curves of different methods at different training episodes. Due to the large action space in the last three datasets, there is no much performance increase for EAR during the online training process, even getting worse. Besides, the curves of SCPR and SeqCR are very vibrant, since they only consider the policy of when to ask or recommend, while the decisions of question-asking and recommendation are made by two separated components. HMA-UNICORN, IMA-UNICORN, and UNICORN preserve more stable training process than other baselines. Among these three methods, UNICORN can converge with the least number of training episodes, i.e., interaction data. HMA-UNICORN can achieve the best performance with an acceptable number of training episodes, while IMA-UNICORN can be regarded as a trade-off model between performance and training efficiency.

6.3 Comparison at Different Conversation Turns

6.3.1 Success Rate at Different Conversation Turns

Besides SR@15, we also present the performance comparison of success rate at each turn (SR@t) in Fig. 3. 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) UNICORN, IMA-UNICORN, and HMA-UNICORN substantially and consistently outperform other baselines across all the datasets and almost every conversation turn.
ii) UNICORN, IMA-UNICORN, and HMA-UNICORN achieve 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 the proposed methods to effectively handle large candidate action space in different situations.
iii) The performance of SCPR gets closer to UNICORN at the latter stage of the conversation, as the candidate action space is getting smaller and the task becomes easier.
iv) 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.
v) Compared with UNICORN, IMA-UNICORN and HMA-UNICORN achieve better performance than UNICORN on the LastFM* dataset. Besides, the performance of IMA-UNICORN and HMA-UNICORN is also getting better at the latter stage of the conversation on LastFM, YELP*, and E-Commerce datasets.

6.3.2 Ratio and Success Rate of Asking

In order to analyze how HMA-UNICORN improves the performance of UNICORN, we further conduct experiments on the ratio of selecting action "ask" and the success rate of question prediction at each turn. As shown in Fig. 4, it can be observed that, generally, HMA-UNICORN tends to ask questions at the beginning of the conversation, while UNICORN decides to ask questions with a relatively random ratio at different turns of the conversation. Especially on those applications with a large number of candidate items (e.g., YELP* and E-Commerce), HMA-UNICORN is more likely to ask questions first for reducing the number of candidate items, with a larger ratio of asking questions at around 2rd-8th turn. This explains the reason why the relative success rate of HMA-UNICORN against UNICORN is getting better at the last few turns of the conversation. On those applications with a large number of candidate

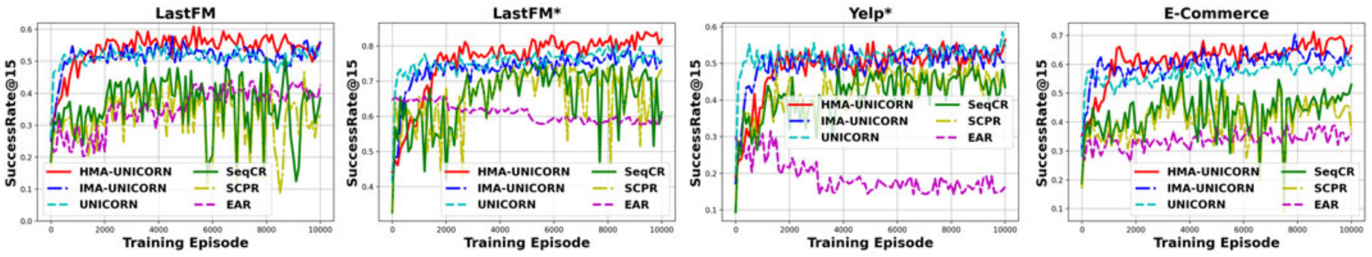

Fig. 2. Test performance at different training episodes.

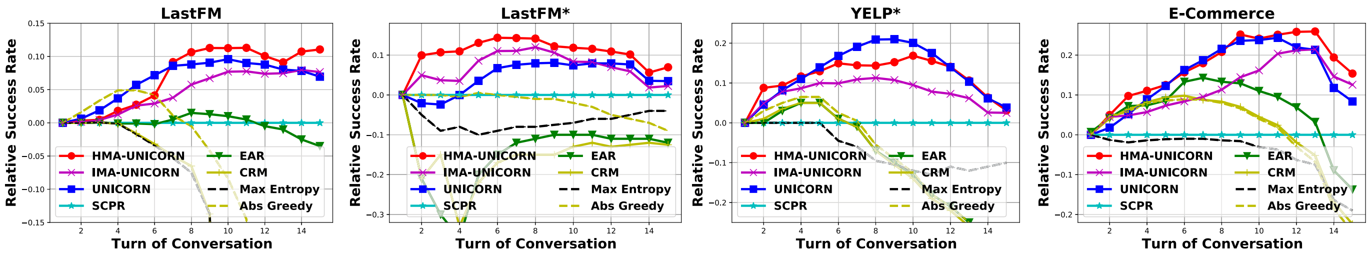

Fig. 3. Comparisons of success rate at different conversation turns.

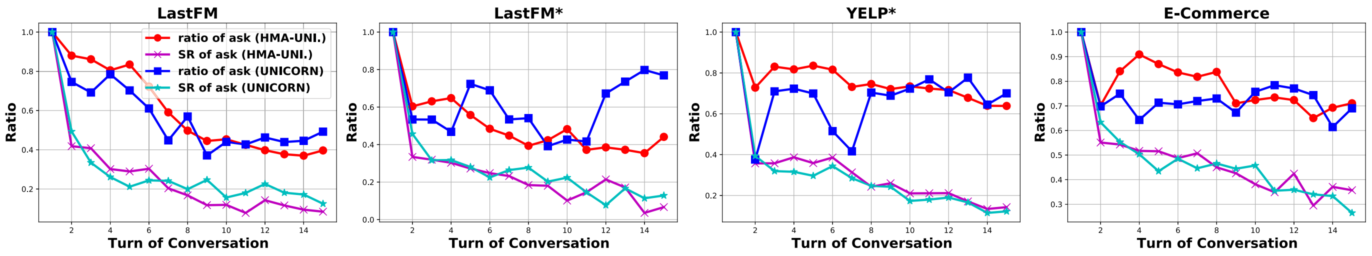

Fig. 4. Ratio and success rate of asking clarification question at different conversation turns.

attributes to be asked (e.g., LastFM*, YELP*, and E-Commerce), UNICORN still tends to ask questions with a higher ratio at the last few turns of the conversation than HMA-UNICORN. These results show that UNICORN falls short of making a good decision on “when to ask or recommend”. Due to the remarkable scalability of the MARL framework, HMA-UNICORN can be adapted to different applications by incorporating global information to adjust the decision-making.

6.4 Detailed Analyses & Discussions

6.4.1 *Components of Attribute Prediction Agent*

The first part in Table 3 (row (a-d)) presents the results that we replace or discard the attribute selection strategies and discard the offline user preference information. 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 [2]. The results (row (a, b)) show that the performance 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 (row (c)), 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*). Since there is no such kind of data like user-liked attributes in Yelp datasets, we only report the results on LastFM and E-Commerce datasets (row(d)). It can be observed that the user-liked attributes contribute to a higher success rate and a fewer average turn, which validates the importance of combining online and offline user preferences in determining what questions to ask.

6.4.2 *Components of Item Recommendation Agent*

The second part in Table 3 (row (e-f)) presents the results of that we discard the item selection or the user’s historical interaction information. We observe that HMA-UNICORN is merely working without item selection, since there are no pre-trained recommendation components in the framework and the preference-based item selection serves as an auxiliary item recall process. Besides, the performance is improved by incorporating the historical interaction across all the datasets. However, the information of user-liked attributes is more effective in LastFM* and E-Commerce datasets than the information of historical interactions, which indicates that it attaches more importance to consider long-term user preference on attributes when there is a large number of attributes to ask.

6.4.3 *Features of Meta State*

In HMA-UNICORN, the high-level meta agent is responsible for coordinating the decisions made by two low-level agents. As described in Section 4.5.1, we consider four features in the meta state representations, including the user’s

TABLE 3
Ablation Study

	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
HMA-UNICORN	**0.576**	**11.35**	**0.185**	**0.824**	**6.78**	**0.421**	**0.986**	**5.02**	**0.402**	0.512	**11.14**	**0.205**	**0.672**	**10.40**	**0.249**
Attribute Prediction Agent															
(a) - Preference-based Attr. Sel.	0.498	12.12	0.159	0.762	7.71	0.343	0.976	5.69	0.381	0.493	11.56	0.174	0.601	11.01	0.211
(b) - Entropy-based Attr. Sel.	0.538	11.60	0.176	0.796	7.02	0.396	0.983	5.44	0.392	0.504	11.38	0.188	0.634	10.71	0.226
(c) - w/o Attribute Selection	0.501	12.01	0.166	0.633	9.21	0.292	0.951	6.66	0.359	0.189	13.27	0.051	0.465	11.78	0.140
(d) - w/o Long-term Preference	0.540	11.84	0.172	0.792	7.24	0.359	-	-	-	-	-	-	0.632	10.88	0.213
Item Recommendation Agent															
(e) - w/o Item Selection	0.161	13.79	0.059	0.658	8.67	0.298	0.801	9.02	0.304	0.152	13.71	0.049	0.169	13.52	0.072
(f) - w/o Long-term Preference	0.531	11.95	0.154	0.792	7.13	0.362	0.978	5.34	0.388	0.501	11.43	0.186	0.652	10.77	0.226
Meta Agent															
(g) - w/o s_{his}	0.566	11.54	0.181	0.816	6.85	0.417	0.986	5.12	0.398	**0.515**	11.16	0.203	0.661	10.72	0.234
(h) - w/o s_{len}	0.550	11.79	0.162	0.796	7.11	0.371	0.980	5.29	0.388	0.508	11.17	0.203	0.666	10.51	0.239
(i) - w/o APA Q-values	0.542	11.74	0.178	0.799	7.12	0.370	0.975	5.56	0.390	0.489	11.66	0.165	0.643	10.97	0.218
(j) - w/o IRA Q-values	0.539	11.80	0.172	0.804	7.01	0.379	0.971	5.60	0.387	0.482	11.82	0.158	0.650	10.86	0.221

SR and hDCG are the higher the better, while AT is the lower the better.

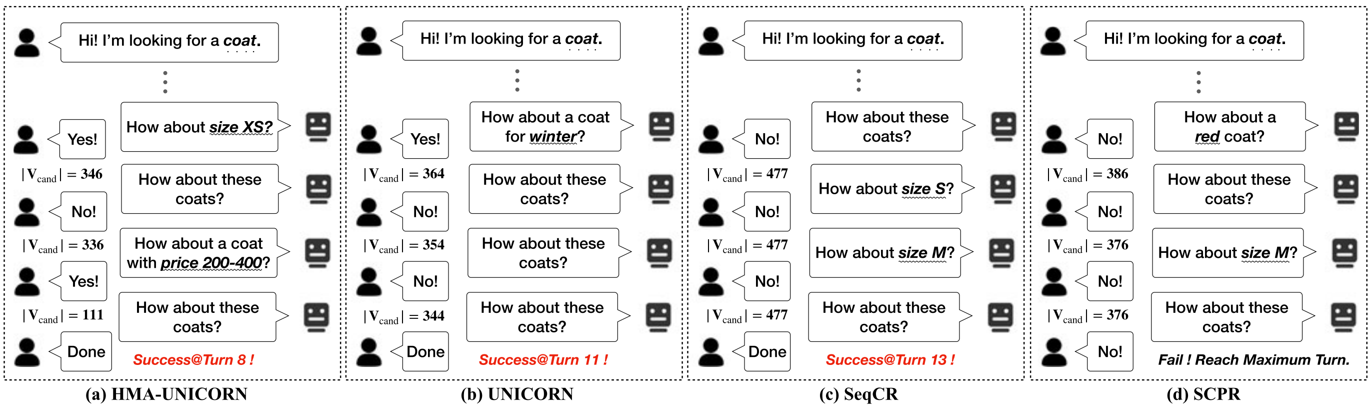

Fig. 5. Sample conversations generated by HMA-UNICORN, UNICORN, SeqCR, and SCPR. For simplicity, we only show the conversation at the last four turns. $|V_{\text{cand}}|$ denotes the number of candidate items at the current turn.

feedback history s_{his}, the size of candidate items s_{len}, and the Q-values of the selected actions by two low-level agents, i.e., $Q^p(s_t, a_t^p)$ and $Q^v(s_t, a_t^v)$. Here we investigate the effect of each feature by discarding them in the meta state representation. The third part in Table 3 (row (g-j)) summarizes the ablation results. As discussed in Section 6.3.2, the incorporation of user's feedback history and the size of candidate items make HMA-UNICORN tend to ask questions at the beginning of the conversation or when there is an extremely large candidate item set. This also provides the evidence for explaining why HMA-UNICORN can achieve a better performance than IMA-UNICORN. Here we observe that both of these two features contribute to a substantial improvement (row (g-h) across all the metrics and datasets. In addition, the Q-values of two low-level agents also attach great importance in deciding the final action (row (i-j)).

6.5 Case Study

In order to intuitively study the difference between the proposed HMA-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 HMA-UNICORN, UNICORN, SeqCR, and SCPR with the user simulator are presented in Fig. 5. With the global features (e.g., the user feedback sequence), HMA-UNICORN, SeqCR, and SCPR can adjust the decision of "when to ask or recommend" when encountering the failure, instead of continuously making recommendations in UNICORN. Compared with SCPR, SeqCR incorporates users' historical interaction information into the recommendation model for modeling the long-term user preference, which leads to better performance on recommendation. HMA-UNICORN makes a better decision of the next action by sys-

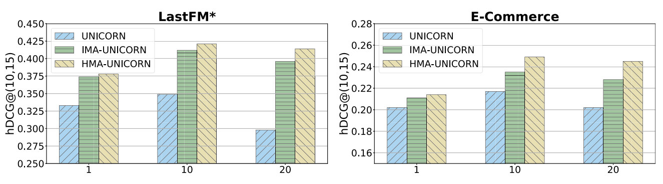

Fig. 6. Effect of the number of selected candidate attributes.

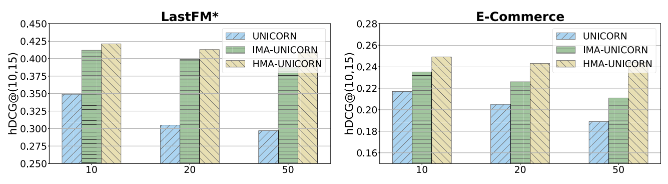

Fig. 7. Effect of the number of selected candidate items.

tematically combining the merits of global features and long short-term user preference modeling.

6.6 Parameter Sensitivity Analysis

Fig. 6 and Fig. 7 present the experimental results (hDCG) by varying the number of selected candidate actions. 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), all these three methods generally achieve the best performance when only selecting a small number of candidate items for policy learning (e.g., K_v=10). The results also demonstrate the necessity of pruning the available actions when there is a large action search space in UCRPL. As for IMA-UNICORN and HMA-UNICORN, the increase of the number of selected actions (K_p or K_v) casts less negative impact on these two methods than that on UNICORN, which indicates that the incorporation of long-term user preference information enhances the robustness of policy learning with more candidate actions.

7 Conclusions and Future Work

In this paper, we propose two novel multi-agent RL-based frameworks, namely IMA-UNICORN and HMA-UNICORN, which utilize different important features and facilitate better unified policy learning in multi-round conversational recommendation systems. In specific, two low-level agents, including attribute prediction agent and item recommendation agent, enrich the state representations by combining the long-term user preference information from the historical interaction data and the short-term user preference information from the conversation history. A high-level meta agent is responsible for coordinating the low-level agents to adaptively make the final decision. Experimental results on four public CRS datasets and a real-world E-Commerce application show that the proposed methods significantly outperform state-of-the-art methods. Extensive analyses further demonstrate the superior scalability of the MARL frameworks on MCR.

This work is the first attempt of applying multi-agent reinforcement learning to conversational recommendation systems. There are some limitations and room for further improvement. For example, we only consider some heuristic reward functions for simplifying the MARL framework, while it would be better to adopt other sophisticated reward functions or extend to actor-critic based frameworks. As for the MARL framework itself, it would also be valuable to investigate the cooperation and competition organisms in CRS, which, in return, can benefit better policy learning in the multi-round CRS. From another perspective, the quality of the user simulator is important in the applications of reinforcement learning methods on CRS. Therefore, it is also worth studying more natural user simulation approaches that can be better correlated with real users.

Acknowledgments

Partial work was done when Yang Deng was an intern at Alibaba.

References

[1] Y. Zhang, X. Chen, Q. Ai, L. Yang, and W. B. Croft, "Towards conversational search and recommendation: System ask, user respond," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, 2018, pp. 177–186.

[2] W. Lei et al., "Estimation-action-reflection: Towards deep interaction between conversational and recommender systems," in *Proc. 13th Int. Conf. Web Search Data Mining*, 2020, pp. 304–312.

[3] R. Li, S. E. Kahou, H. Schulz, V. Michalski, L. Charlin, and C. Pal, "Towards deep conversational recommendations," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 9748–9758.

[4] S. Rendle, "Factorization machines," in *Proc. IEEE Int. Conf. Des. Mater.*, 2010, pp. 995–1000.

[5] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 173–182.

[6] W. X. Zhao et al., "Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms," *CoRR*, 2020, *arXiv:2011.01731*.

[7] L. Zou et al., "Pseudo dyna-Q: A reinforcement learning framework for interactive recommendation," in *Proc. 13th Int. Conf. Web Search Data Mining*, 2020, pp. 816–824.

[8] S. Zhou et al., "Interactive recommender system via knowledge graph-enhanced reinforcement learning," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 179–188.

[9] W. Lei, X. He, M. de Rijke, and T. Chua, "Conversational recommendation: Formulation, methods, and evaluation," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 2425–2428.

[10] K. Christakopoulou, F. Radlinski, and K. Hofmann, "Towards conversational recommender systems," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 815–824.

[11] J. Zou, Y. Chen, and E. Kanoulas, "Towards question-based recommender systems," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 881–890.

[12] Y. Sun and Y. Zhang, "Conversational recommender system," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2018, pp. 235–244.

[13] W. Lei et al., "Interactive path reasoning on graph for conversational recommendation," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 2073–2083.

[14] K. Zhou, Y. Zhou, W. X. Zhao, X. Wang, and J. Wen, "Towards topic-guided conversational recommender system," in *Proc. 29th Int. Conf. Comput. Linguistics*, 2020, pp. 4128–4139.

[15] K. Christakopoulou, A. Beutel, R. Li, S. Jain, and E. H. Chi, "Q&R: A two-stage approach toward interactive recommendation," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 139–148.

[16] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 1999, pp. 1057–1063.

[17] X. Tian, Y. Hao, P. Zhao, D. Wang, Y. Liu, and V. S. Sheng, "Considering interaction sequence of historical items for conversational recommender system," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, 2021, pp. 115–131.

[18] Y. Deng, Y. Li, F. Sun, B. Ding, and W. Lam, "Unified conversational recommendation policy learning via graph-based reinforcement learning," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2021, pp. 1431–1441.

[19] K. Zhou et al., "Leveraging historical interaction data for improving conversational recommender system," in *Proc. Conf. Inf. Knowl. Manage.*, 2020, pp. 2349–2352.

[20] Y. Zhu et al., "What to do next: Modeling user behaviors by time-LSTM," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, 2017, pp. 3602–3608.

[21] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Comput. Surv.*, vol. 52, no. 1, 2019.

[22] C. Gao, W. Lei, X. He, M. de Rijke, and T. Chua, "Advances and challenges in conversational recommender systems: A survey," 2021, *arXiv:2101.09459*.

[23] X. Zhang, H. Xie, H. Li, and J. C. S. Lui, "Conversational contextual bandit: Algorithm and application," in *Proc. Web Conf.*, 2020, pp. 662–672.

[24] S. Li, W. Lei, Q. Wu, X. He, P. Jiang, and T.-S. Chua, "Seamlessly unifying attributes and items: Conversational recommendation for cold-start users," *ACM Trans. Inf. Syst.*, vol. 39, pp. 1–29, 2021.

[25] L. Wang, H. Hu, L. Sha, C. Xu, K. Wong, and D. Jiang, "Finetuning large-scale pre-trained language models for conversational recommendation with knowledge graph," 2021, *arXiv:2110.0747*.
[26] Y. Deng, W. Zhang, W. Xu, W. Lei, T. Chua, and W. Lam, "A unified multi-task learning framework for multi-goal conversational recommender systems," 2022, *arXiv:2204.06923*.
[27] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," in *Proc. Int. Conf. Learn. Representations*, 2016.
[28] O. Sakhi, S. Bonner, D. Rohde, and F. Vasile, "BLOB: A probabilistic model for recommendation that combines organic and bandit signals," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 783–793.
[29] G. Zheng et al., "DRN: A deep reinforcement learning framework for news recommendation," in *Proc. World Wide Web Conf.*, 2018, pp. 167–176.
[30] X. Zhao, L. Zhang, Z. Ding, L. Xia, J. Tang, and D. Yin, "Recommendations with negative feedback via pairwise deep reinforcement learning," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 1040–1048.
[31] G. Shani, D. Heckerman, and R. I. Brafman, "An mdp-based recommender system," *J. Mach. Learn. Res.*, vol. 6, pp. 1265–1295, 2005.
[32] C. Pei et al., "Value-aware recommendation based on reinforcement profit maximization," in *Proc. World Wide Web Conf.*, 2019, pp. 3123–3129.
[33] X. Xin, A. Karatzoglou, I. Arapakis, and J. M. Jose, "Self-supervised reinforcement learning for recommender systems," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 931–940.
[34] P. Wang, Y. Fan, L. Xia, W. X. Zhao, S. Niu, and J. Huang, "KERL: A knowledge-guided reinforcement learning model for sequential recommendation," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 209–218.
[35] R. Zhang, T. Yu, Y. Shen, H. Jin, and C. Chen, "Text-based interactive recommendation via constraint-augmented reinforcement learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 15 188–15 198.
[36] X. Wang, X. He, M. Wang, F. Feng, and T. Chua, "Neural graph collaborative filtering," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2019, pp. 165–174.
[37] L. Zheng, C. Lu, F. Jiang, J. Zhang, and P. S. Yu, "Spectral collaborative filtering," in *Proc. 12th ACM Conf. Recommender Syst.*, 2018, pp. 311–319.
[38] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "LightGCN: Simplifying and powering graph convolution network for recommendation," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 639–648.
[39] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W. Ma, "Collaborative knowledge base embedding for recommender systems," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 353–362.
[40] J. Huang, W. X. Zhao, H. Dou, J. Wen, and E. Y. Chang, "Improving sequential recommendation with knowledge-enhanced memory networks," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2018, pp. 505–514.
[41] W. Ma et al., "Jointly learning explainable rules for recommendation with knowledge graph," in *Proc. World Wide Web Conf.*, 2019, pp. 1210–1221.
[42] X. Wang, D. Wang, C. Xu, X. He, Y. Cao, and T. Chua, "Explainable reasoning over knowledge graphs for recommendation," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 5329–5336.
[43] X. Wang, Y. Xu, X. He, Y. Cao, M. Wang, and T. Chua, "Reinforced negative sampling over knowledge graph for recommendation," in *Proc. Web Conf.*, 2020, pp. 99–109.
[44] Y. Xian, Z. Fu, S. Muthukrishnan, G. de Melo, and Y. Zhang, "Reinforcement knowledge graph reasoning for explainable recommendation," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2019, pp. 285–294.
[45] Y. Lei, H. Pei, H. Yan, and W. Li, "Reinforcement learning based recommendation with graph convolutional Q-network," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 1757–1760.
[46] K. Zhao et al., "Leveraging demonstrations for reinforcement recommendation reasoning over knowledge graphs," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 239–248.
[47] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2017.
[48] L. Busoniu, R. Babuska, and B. D. Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Syst. Man Cybern. Part C*, vol. 38, no. 2, pp. 156–172, Feb. 2008.
[49] M. Tan, "Multi-agent reinforcement learning: Independent versus cooperative agents," in *Proc. 10th Int. Conf. Int. Conf. Mach. Learn.*, 1993, pp. 330–337.
[50] J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 2137–2145.
[51] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2961–2970.
[52] X. He et al., "Learning to collaborate in multi-module recommendation via multi-agent reinforcement learning without communication," in *Proc. 14th ACM Conf. Recommender Syst.*, 2020, pp. 210–219.
[53] J. Feng et al., "Learning to collaborate: Multi-scenario ranking via multi-agent reinforcement learning," in *Proc. World Wide Web Conf.*, 2018, pp. 1939–1948.
[54] W. Zhang et al., "Intelligent electric vehicle charging recommendation based on multi-agent reinforcement learning," in *Proc. Web Conf.*, 2021, pp. 1856–1867.
[55] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 2787–2795.
[56] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Proc. 28th AAAI Conf. Artif. Intell.*, 2014, pp. 1112–1119.
[57] A. Vaswani et al., "Attention is all you need," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
[58] W. Kang and J. J. McAuley, "Self-attentive sequential recommendation," in *Proc. IEEE Int. Conf. Des. Mater.*, 2018, pp. 197–206.
[59] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[60] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling network architectures for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1995–2003.
[61] R. Bellman and R. Kalaba, "On the role of dynamic programming in statistical communication theory," *IRE Trans. Inf. Theory*, vol. 3, no. 3, pp. 197–203, 1957.
[62] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 2094–2100.
[63] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proc. Int. Conf. Learn. Representations*, 2016.
[64] X. Han et al., "OpenKE: An open toolkit for knowledge embedding," in *Proc. Conf. Empirical Methods Natural Lang. Process.: Syst. Demonstrations*, 2018, pp. 139–144.

Yang Deng received the BS degree from the Beijing University of Posts and Telecommunications and the MS degree from Peking University. He is now working toward the PhD degree in the Department of System Engineering and Engineering Management, The Chinese University of Hong Kong. His research interests include Natural Language Processing, Information Retrieval, and Deep Learning.

Yaliang Li received the PhD degree from the Department of Computer Science and Engineering, SUNY Buffalo, in 2017. He is a research scientist with DAMO Academy, Alibaba Group. Before that he worked as a research scientist with Baidu Research, and a senior researcher with Tencent Medical AI Lab. He is broadly interested in machine learning and data mining with a focus on truth discovery, knowledge graph, question answering, differential privacy, recommendation, and more recently automated machine learning.

Bolin Ding received the BS degree in math and applied mathematics with the Renmin University of China, the MPhil degree in systems engineering and engineering nanagement with the Chinese University of Hong Kong, and the PhD degree in computer science with the University of Illinois, Urbana-Champaign. He is a research scientist in the Data Analytics and Intelligence Lab (DAIL) with Alibaba Group. Prior to joining Alibaba, he worked as a researcher in Microsoft Research. His research focuses on the management and analytics of large-scale data, including real-time approximate query algorithms and systems, data privacy protection, query processing and optimization algorithms, and algorithms and applications of data mining and machine learning. He has hold more than 10 US patents. He received the 2017 Technical Excellence Award from Microsoft Privacy for his contributions on the research and deployment of data privacy techniques.

Wai Lam (Senior Member, IEEE) received the BSc and the MPhil degrees from the Chinese University of Hong Kong, the PhD degree, he conducted research with Indiana University Purdue University Indianapolis (IUPUI) and the University of Iowa, and the PhD degree in computer science from the University of Waterloo. He joined The Chinese University of Hong Kong, where he is currently a professor. His research interests include text mining, natural language processing, and intelligent information retrieval. He has published extensively in top-tier conferences and journals in these areas. His research projects have been funded by the Hong Kong SAR Government General Research Fund (GRF), Alibaba Group, Huawei, and DARPA. He also managed industrial projects funded by Innovation and Technology Fund (industrial grant) and IT companies.