# Chain-of-Choice Hierarchical Policy Learning for Conversational Recommendation

Wei Fan[1], Weijia Zhang[2], Weiqi Wang[1], Yangqiu Song[1], and Hao Liu[2(✉)]

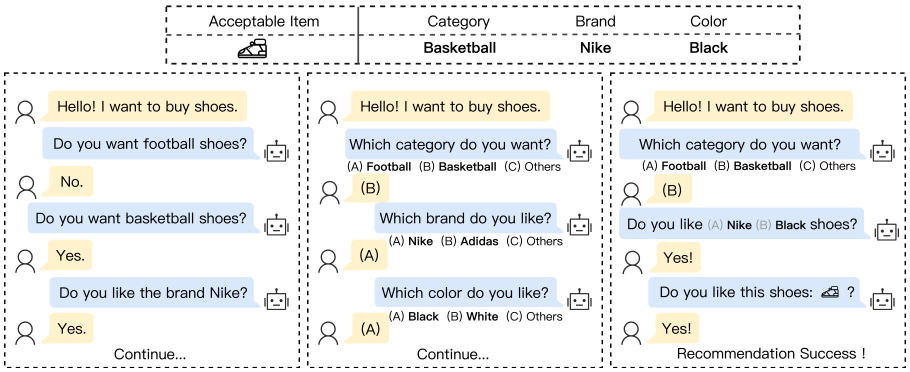
[1] The Hong Kong University of Science and Technology, Hong Kong, China
wfanag@connect.ust.hk, {wwangbw, yqsong}@cse.ust.hk
[2] The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China
wzhang411@connect.hkust-gz.edu.cn, liuh@ust.hk

**Abstract.** Conversational Recommender Systems (CRS) illuminate user preferences via multi-round interactive dialogues, ultimately navigating towards precise and satisfactory recommendations. However, contemporary CRS are limited to inquiring binary or multi-choice questions based on a single attribute type (*e.g.*, color) per round, which causes excessive rounds of interaction and diminishes the user's experience. To address this, we propose a more realistic and efficient conversational recommendation problem setting, called **M**ulti-**T**ype-**A**ttribute **M**ulti-round **C**onversational **R**ecommendation (**MTAMCR**), which enables CRS to inquire about multi-choice questions covering multiple types of attributes in each round, thereby improving interactive efficiency. Moreover, by formulating MTAMCR as a hierarchical reinforcement learning task, we propose a **C**hain-**o**f-**C**hoice **H**ierarchical **P**olicy **L**earning (**CoCHPL**) framework to enhance both the questioning efficiency and recommendation effectiveness in MTAMCR. Specifically, a long-term policy over options (*i.e.*, ask or recommend) determines the action type, while two short-term intra-option policies sequentially generate the chain of attributes or items through multi-step reasoning and selection, optimizing the diversity and interdependence of questioning attributes. Finally, extensive experiments on four benchmarks demonstrate the superior performance of CoCHPL over prevailing state-of-the-art methods.

**Keywords:** Conversational Recommendation · Hierarchical Reinforcement Learning · Graph Representation Learning.

## 1 Introduction

Compared to traditional recommender systems, conversational recommender system (CRS) [12] serves as an online salesperson who elicits user preferences [11,17,9] through engaging question-and-answer dialogues [6,20] to provide tailored recommendations. This conversational approach allows for explicit preference reasoning by actively asking questions about attributes and making desirable recommendations based on the user's real-time feedback [7].




(a) Binary choices (b) Single-Type attributes (c) Multi-Type attributes

**Fig. 1.** Example of different conversational recommendation settings.

Recent research has explored different settings to create more realistic conversational recommendation scenarios. Existing studies allow CRS to ask either binary (yes/no) questions for each selected attribute [12,6,8,4] or questions with multiple choices falling into one selected attribute type (*e.g.*, brand) in each round [19]. However, existing CRS using binary choices [4] or single-type attribute questions [19] can be repetitive and time-consuming, causing frustration for the user and decreasing the overall recommendation success rate. Consider an example illustrated in Figure 1, where the user wants shoes with the attributes "Basketball," "Nike," and "Black." In Figure 1(a) and Figure 1(b), the question choices are restricted to one attribute type per round, necessitating at least three rounds to extract all the preferred attributes. Our work studies a more realistic scenario, namely Multi-Type-Attribute Multi-round Conversational Recommendation (MTAMCR) as depicted in Figure 1(c), which enables CRS to generate questions encompassing diverse yet dependent attribute types. By optimizing the combination of attribute types, we can enhance the efficiency of reasoning and questioning, ultimately increasing the success rate of recommendations.

Previous works [12,6,8,4,19,21] have formulated conversational recommendation as a Markov Decision Process (MDP) and introduced reinforcement learning (RL) to choose the attributes and recommended items. However, these approaches encounter two main challenges in the MTAMCR scenario: 1) Diversity of attributes: Selecting the attribute instances within a single attribute type only allows for exploring one user's attribute preference (*e.g.*, color) per round. Thus, these methods need more rounds to discover and capture all the attributes of the user's target item. 2) Dependency between attributes: Directly selecting attribute instances based on their top-K Q-values only considers the relationship between attributes and the current environment state. However, they neglect the influence of previous attribute choices on subsequent attribute selections, leading to suboptimal combinations of attributes.

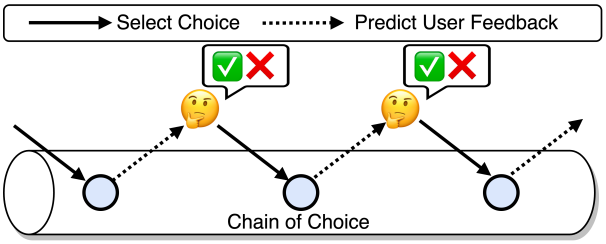



**Fig. 2.** During each turn, the agent engages in a decision-making process where it selects a choice and then predicts the user's feedback in order to infer the subsequent state. With the predicted state, the agent selects the next choice, continually repeating this process until the round eventually reaches its termination point.

To address the two challenges, we remove the restriction on attribute types within each round and propose the chain-of-choice generation approach, as illustrated in Figure 2. This approach infers the next choice, considering the previous choices by predicting the user feedback. For example, if the user wants basketball shoes and the first choice in a given turn is "Nike," selecting the color "Black" as the next choice is more effective than presenting a brand selection between "Nike" and "Adidas." This is because we predict that the user would accept the attribute "Nike" since many individuals highly prefer Nike basketball shoes. Therefore, it is strategically advantageous to prioritize exploring color preferences rather than spending more time deliberating over brand preferences. With the chain of choices, we first proposed the Chain-of-Choice Hierarchical Policy Learning (CoCHPL) framework, which formulates MTAMCR as a hierarchical reinforcement learning task. In each turn, CoCHPL employs a long-term policy to select the option (*i.e.*, ask or recommend) and generates the chain of choices (*i.e.*, attributes or items) step by step via different short intra-option policies. To model the transitions of the choice chain and determine the optimal moment for terminating the current choice generation, we utilize learnable functions that predict feedback for the ongoing choice chain and infer the next state for the next choice selection.

Our major contributions can be summarized as follows:

1) We introduce the **MTAMCR** scenario, which allows the CRS agent to ask questions under the same or different attribute types in each round, enabling more realistic and effective user interaction.
2) We propose the **CoCHPL** framework for the MTAMCR scenario by formulating MTAMCR as a hierarchical RL task, which includes an option selection task and chain-of-choice generation tasks for different options (action types).
3) We conduct extensive experiments on four benchmark datasets, and the results demonstrate a substantial improvement in both performance and generative capacity with all the comparison methods.

## 2 Related Work

### 2.1 Multi-round Conversational Recommendation

The most common setting for conversational recommendation is multi-round conversational recommendation (MCR) [4,8,18], which allows for multiple rounds of questioning and recommendations until user acceptance or a maximum number of dialogue rounds is reached. In MCR, the system focuses on two tasks: 1) asking appropriate questions about attributes to extract user preferences and 2) recommending items that the user is most likely to choose. CRM [12] introduced reinforcement learning to select actions in the single-round scenario, and EAR [6] extended it to the MCR problem. SCPR [8] models conversational recommendation as an interactive path reasoning problem on a graph, and UNICORN [4] proposes a unified framework based on a dynamic weighted graph. DAHCR [21] introduces a hierarchical framework in which one policy selects the action types, and another policy selects the attributes or items. Furthermore, MCMIPL [19] proposes Multiple-Choice MCR, allowing CRS to ask multiple-choice questions within one attribute type instead of one question. However, previous works limit themselves to asking single-type attribute questions in each turn. To overcome this limitation, we develope a more realistic setting named Multi-Type-Attribute Multi-round Conversational Recommendation.

### 2.2 The Options Framework

The options framework, introduced by Sutton et al. [14,10], augments the conventional reinforcement learning (RL) paradigm, integrating a set of temporally extended actions and termed options. In our research, each option $\omega \in \Omega$ is characterized as a unique action type embodying an intra-option policy $\pi_\omega$ for action/instance selection, a prediction function $\mathcal{T}_\omega$ for predicting the subsequent state, and a termination function $\beta_\omega$ to decide when to terminate the current option. The framework provides a hierarchical structure, enabling efficient exploration and decomposition of complex tasks into manageable sub-tasks. Based on the options framework, option-critic, proposed by Bacon et al. [1], combines the benefits of the options framework with deep neural networks and policy gradient methods [13]. It heralds a pioneering architecture that concurrently learns intra-option policies and termination conditions, thereby facilitating the identification of subgoals without the prerequisite of additional reward configurations.

## 3 Preliminary

**Multi-Type-Attribute Multi-round Conversational Recommendation.** Following previous works [6,8,19], we assume that the user preserves clear preferences toward attributes and items. CRS engages in a dialogue with users, asking questions about attributes or providing recommendations in each round. Users can express their preferences by accepting or rejecting the attributes or

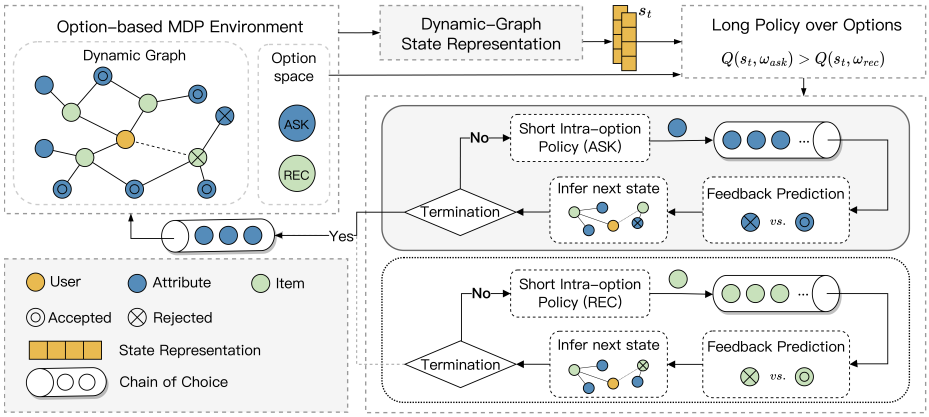



**Fig. 3.** The overview of Chain-of-Choice Hierarchical Policy Learning.

items mentioned by the CRS. The conversation continues until either a successful recommendation is made or the maximum number of rounds is reached. Furthermore, we focus on a more realistic CRS setting, namely Multi-Type-Attribute Multi-round Conversational Recommendation (MTAMCR), where the CRS can question the user about multiple types of attributes in each turn.

Specifically, we define the user set as $\mathcal{U}$, the item set as $\mathcal{V}$, and the attribute set as $\mathcal{P}$. Each item $v \in \mathcal{V}$ consists of a set of attribute instances $\mathcal{P}_v$, where each attribute instance $p \in \mathcal{P}_v$ belongs to a specific attribute type. In each episode, there is a target item $v$ that is acceptable to the user $u \in \mathcal{U}$. At the beginning of the episode, the user $u$ provides a preferred attribute $p_0 \in \mathcal{P}_v$ to the CRS. Then, in each turn $T$, the CRS is free to ask multiple choice questions:

$$C_T = \{a_1, a_2, ..., a_n\}, \tag{1}$$

where $a$ represents a choice (*i.e.*, attribute or item), and $C_T$ can contain multiple attributes $\{p_1, p_2, ..., p_n\}$ or multiple items $\{v_1, v_2, ..., v_n\}$. The user $u$ can accept or reject each choice in $C_T$ asked by the CRS.

## 4 Chain-of-Choice Hierarchical Policy Learning

In this section, we present a detailed description of our method, CoCHPL, as illustrated in Figure 3, to explain the overview of the chain-of-choice generation. CoCHPL consists of four key components: an option-based MDP Environment to provide the dynamic state with the graph structure and candidate options and choices for the agent, a Dynamic-Graph State Representation module that encodes the graph-based state into a latent space to capture the key information and relationships, a Long Policy Over Options to determine the option (*i.e.*, ask or recommend) per round, and a Short Intra-Option Policy to generate the chain of choices via multi-step reasoning and selection.

### 4.1 Option-based MDP Environment

By formulating the MTAMCR as a hierarchical reinforcement learning task, we model not only the low-level actions (choices) but also the high-level actions (options) within each turn, which helps the CRS learn different dimensions of temporal abstractions. The original environment consists of a state space $S$, an action space $\mathcal{A}$, a transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ and a reward funtion $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The Option-Based MDP environment extends the MDP environment to the tuple $(\mathcal{S}, \Omega, \mathcal{A}, \mathcal{T}, \mathcal{R})$ by introducing options $\Omega = \{\omega_{ask}, \omega_{rec}\}$, which are alternative action types that the agent can choose from at each turn.

Furthermore, to model the state transitions between the multiple choices in the same turn, we assume that the user responds to the choices one by one and introduce the timestep $t$ to expand the state of turn $T$ as follows:

$$s_T = \{s_t, s_{t+1}, ..., s_{t+n}\}, s_{T+1} = \{s_{t+n}, s_{t+n+1}, ..., s_{t+n+m}\}, \tag{2}$$

where $n$, $m$ is the choice number in turn $T$ and $T+1$, respectively. If the user responds to the first choice $a_t$ at turn $T$, the state will transition from $s_t$ to $s_{t+1}$.

**State-Option Space** Before each turn $T$, the CRS selects an option $\omega_T$ from $\Omega$ to decide the action type and extend the state space. The state-option pair $(s_t, \omega_T)$ at timestep $t$ in turn $T$ contains $\{u, \mathcal{V}^{(t)}, \mathcal{P}^t, \omega_T\}$, where $\mathcal{V}^{(t)} = \mathcal{V}^{(t)}_{rej} \cup \mathcal{V}^{(t)}_{cand}$, and $\mathcal{P}^{(t)} = \mathcal{P}^{(t)}_{acc} \cup \mathcal{P}^{(t)}_{rej} \cup \mathcal{P}^{(t)}_{cand}$. $\mathcal{P}^{(t)}_{acc}$ denotes all the accepted attributes, $\mathcal{V}^{(t)}_{rej}$ and $\mathcal{P}^{(t)}_{rej}$ denotes all the rejected items and attributes, $\mathcal{V}^{(t)}_{cand}$ and $\mathcal{P}^{(t)}_{cand}$ denotes the candidate items and attributes.

**Action Space.** For $(s_t, \omega_{rec})$ at timestep $t$, the candidate action space $\mathcal{A}_t$ is $\mathcal{V}^{(t)}_{cand}$ for the option $\omega_{rec}$. We have $\mathcal{V}^{(t)}_{cand} = \mathcal{V}_{\mathcal{P}^{(t)}_{acc}} \backslash \mathcal{V}^{(t)}_{rej}$ where $\mathcal{V}_{\mathcal{P}^{(t)}_{acc}}$ denotes all the items which contain all the accepted attributes. For another pair $(s_t, \omega_{ask})$, the candidate attribute space $\mathcal{P}^{(t)}_{cand} = \mathcal{P}_{\mathcal{V}^{(t)}_{cand}} \backslash (\mathcal{P}^{(t)}_{acc} \cup \mathcal{P}^{(t)}_{rej})$ and $\mathcal{P}_{\mathcal{V}^{(t)}_{cand}}$ denotes the attributes of all the candidate items.

**Transition.** For the state-option pair $(s_t, \omega_T)$, after the selection of $a_t$ at timestep $t$, if the CRS predicts that the user accepts $a_t$, then $s_t$ will transiton to $s_{t+1}$ and $\mathcal{P}^{(t+1)}_{acc} = \mathcal{P}^{(t)}_{acc} \cup a_t$. If rejected, $\mathcal{P}^{(t+1)}_{rej}$ are updated to $\mathcal{P}^{(t)}_{rej} \cup a_t$. The next candidate action space is updated to $\mathcal{P}^{(t+1)}_{cand} = \mathcal{P}^{(t)}_{cand} \backslash a_t$.

**Reward.** To promote a universal and uniform distribution of rewards $\mathcal{R}$, a simplification of the reward structure is implemented as (1) $r_{ask\_suc}$: a slight reward when the user accepts the asked attribute, (2) $r_{ask\_fail}$: a slight penalty when the user rejects the asked attribute, (3) $r_{rec\_fail}$: a slight penalty when the user rejects the recommended item, (4) $r_{rec\_suc}$: a strong reward when the user accepts the recommended item.

### 4.2 Dynamic-Graph State Representation Learning

We explicitly employ a dynamic-graph state representation in the option-based MDP environment to incorporate user-preferred attributes. This representation captures both the current conversation and its historical context, facilitating the modeling of conversational recommendation as an interactive path reasoning problem on a graph.

**Graph Construction.** We denote the dynamic undirected graph as $G_u = (\mathbf{V}, \mathbf{A})$. The node set $\mathbf{V}$ consists of the related user, items, and attributes. And $\mathbf{A}$ is a $n \times n$ adjacency matrix representing each node's edge weight in $\mathbf{V}$. During each timestep in the conversation, $G_u$ dynamically changes the node and adjacency matrix as follows:

$$\mathbf{V}^{(t)} = \{u\} \cup \mathcal{P}_{acc}^{(t)} \cup \mathcal{P}_{cand}^{(t)} \cup \mathcal{P}_{rej}^{(t)} \cup \mathcal{V}_{cand}^{(t)} \cup \mathcal{V}_{rej}^{(t)}, \tag{3}$$

$$\mathbf{A}_{i,j}^{(t)} = \begin{cases} w_v^{(t)}, & \text{if } n_i = u, n_j \in \mathcal{V}_{cand}^{(t)}, \\ 1, & \text{if } n_i \in \mathcal{V}_{cand}^{(t)}, n_j \in \mathcal{P}, \\ 0, & \text{otherwise }, \end{cases} \tag{4}$$

where $w_v^{(t)}$ denotes the user's preference score for the item $v$, calculated as:

$$w_v^{(t)} = \sigma(e_u^\top e_v + \sum\nolimits_{p \in \mathcal{P}_{acc}^{(t)}} e_v^\top e_p - \sum\nolimits_{p \in \mathcal{P}_{rej}^{(t)}} e_v^\top e_p), \tag{5}$$

where $\sigma(\cdot)$ is the sigmoid function and $e_u, e_v, e_p$ are pre-trained embeddings initialized as [2]. We can also calculate the preference score for candidate attributes at the current state: $w_p^{(t)} = \sigma(e_u^\top e_p + \sum_{p \in \mathcal{P}_{acc}^{(t)}} \overline{e}_p^\top e_p - \sum_{p \in \mathcal{P}_{rej}^{(t)}} \overline{e}_p^\top e_p)$. With the calculated weights, we can reduce the action space size for different options by selecting the top-$K$ candidates in the action selection section 4.3, 4.4.

**State Representation.** By incorporating both the current conversation and the historical conversation data, we can create a comprehensive global knowledge graph denoted as $G$. This knowledge graph captures the evolving preferences of users over time. To effectively leverage the correlation information among users, items, and attributes represented in the graph, we implement a two-layer graph convolutional network (GCN) [5] as follows:

$$e_s^{(l+1)} = \mathrm{ReLU}(\sum\nolimits_{i \in \mathcal{N}_s} D^{-\frac{1}{2}} \boldsymbol{A} D^{-\frac{1}{2}} W^l e_i^{(l)} + B^{(l)} e_s^{(l)}), \tag{6}$$

where $l$ is the layer number, $\mathcal{N}_s$ denotes the neighbors of $n_s$ and $D$ denotes the degree matrix [5] with $D_{ii} = \sum_j \boldsymbol{A}_{i,j}$. $B^{(l)}$ and $W^{(l)}$ are both trainable network parameters. To further extract the sequential information of the current conversation, we use a single-layer multi-head attention [15] network and calculate the final state representation by an average pooling:

$$\mathcal{P}_{acc}^* = \mathrm{LayerNorm}(\mathrm{FNN}(\mathrm{MultiHead}(\mathcal{P}_{acc}^{(t)}) + \mathcal{P}_{acc}^{(t)})) \tag{7}$$

$$f_{\theta_S}(s_t) = \text{AvgPool}(\mathcal{P}^*_{acc}). \tag{8}$$

After the sequential learning, the final state representation is generated, and $\theta_S$ denotes all parameters for the dynamic-graph state representation module.

**Algorithm 1:** CoCHPL in the MTAMCR scenario

Parameter initialization: $\theta_Q$, $\theta_\beta$, $\theta_\mathcal{T}$, $\theta_S$
**for** *episode* $= 1$ **to** $K$ **do**
  Initialization: User $u \in \mathcal{U}$, Target item $v \in \mathcal{V}_u$, Initial attribute $p_0 \in \mathcal{P}_v$
  Initialization: Timestep $t \leftarrow 0$, Turn number $T \leftarrow 0$
  Initialization: Choice number $n \leftarrow 0$, Initial state $s_{t+n} \leftarrow s_0$
  Choose an action option $\omega_T$ via $\epsilon$-soft long policy $\pi_\Omega(s_{t+n})$
  **repeat**
    **1. Chain of Choice Generation:**
    Initialization: choice chain $C_T \leftarrow \{\}$, candidate action space $\mathcal{A}_t$
    **repeat**
      Choose an choice $a_{t+n}$ via short intra-option policy $\pi_{\omega_T}(s_{t+n})$
      **if** *at training stage* **then**
        Take choice $a_{t+n}$, receive true reward $r_{t+n}$
        Observe true $s_{t+n+1}$, get candidate action space $\mathcal{A}_{t+n+1}$
        Store $(s_{t+n}, a_{t+n}, r_{t+n}, s_{t+n+1}, \mathcal{A}_{t+n+1}, \omega_T)$
      **else**
        Predict user feedback and reward via $\mathcal{T}_\omega(s_{t+n}, a_{t+n})$
        Update the next state $s_{t+n+1}$ based on predicted feedback
      **end**
      Update $C_T \leftarrow C_T + a_{t+n},\ n \leftarrow n + 1$
    **until** $\beta_{\omega_T}(s_{t+n})$ *terminates or* $C_T$ *reach the maximum length*;

    **2. User Interaction:**
    Take chain of choice $C_T$, observe $s_{t+n}$
    Choose new option $\omega_{T+1}$ via $\epsilon$-soft $\pi_\Omega(s_{t+n})$
    Update $t \leftarrow t + n$, $n \leftarrow 0$, $T \leftarrow T + 1$

    **3. Policy Optimization:**
    Sample mini-batch of $(s_{t+n}, a_{t+n}, r_{t+n}, s_{t+n+1}, A_{t+n+1}, \omega_T)$
    Update $\theta_Q, \theta_S$ *w.r.t.* Eq. (15)
    Update $\theta_\beta$ *w.r.t.* Eq. (16)
    Update $\theta_\mathcal{T}$ *w.r.t.* Eq. (17)
  **until** *recommended successfully or* $T > T_{max}$;
**end**

### 4.3 Long Policy Leaning Over Options

After encoding the state with the dynamic graph, we propose a long-term policy denoted as $\pi_\Omega$ to select the option $\omega$ at each turn. Our objective is to optimize

the discounted return at turn $T$ directly. To achieve this, we express the expected Q-value of each option as follows:

$$Q_{\Omega}(s_t, \omega_T) = \sum\nolimits_{a_t \in \mathcal{A}_t} \pi_{\omega_T}(a_t \mid s_t) Q_U(s_t, \omega_T, a_t), \tag{9}$$

where $\pi_{\omega_T}$ is the intra-option policy and $Q_U : \mathcal{S} \times \Omega \times \mathcal{A} \to \mathbb{R}$ is the estimated Q-value of executing an action in the context of a state-option pair which is introduced in section 4.4.

### 4.4 Short Intra-Option Policy Learning

With the state-option pair, we need a short-term policy to generate the choice chain at each turn. We propose that each option $\omega$ consists of a triplet $(\pi_\omega, \beta_\omega, \mathcal{T}_\omega)$ in which $\pi_\omega \in \{\pi_{\omega_{ask}}, \pi_{\omega_{rec}}\}$ is a short-term intra-option policy to select the attribute or item at each timestep, $\mathcal{T}_\omega : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is a feedback prediction function to predict user response for inferring the next timestep state, and $\beta_\omega : \mathcal{S} \to [0, 1]$ is a termination function to determine when to exit the generation process of the current option.

As shown in Algorithm 1, during each timestep $t$ in turn $T$, $\pi_\omega$ select a item or attribute as the choice $a_t$, and then predict the feedback (*i.e.*, acc/rej) via $\mathcal{T}_\omega(s_t, a_t) = \mathbb{1}(a_t \text{ is accpeted})$ for inferring the next state $s_{t+1}$. If $\beta_{\omega_T}(s_{t+1})$ determines to terminate the choice chain generation, the CRS will present the chain-of-choice question to the user. Otherwise, the generation will continue.

To optimize intra-option policy $\pi_\omega$, we estimate the Q-value of selecting a choice in the context of a state-option pair $(s, \omega)$, which can be calculated by:

$$Q_U(s_t, \omega_T, a_t) = r(s_t, a_t) + \gamma U(\omega_T, s_{t+1}), \tag{10}$$

where $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function and $U : \Omega \times \mathcal{S} \to \mathbb{R}$ denotes the value function of executing option $\omega_T$ upon entering the next state $s_{t+1}$ as:

$$U(\omega_T, s_{t+1}) = (1 - \beta_{\omega_T}(s_{t+1})) Q_{\Omega}(s_{t+1}, \omega_T) + \beta_{\omega_T}(s_{t+1}) \max_{\overline{\omega}} Q_{\Omega}(s_{t+1}, \overline{\omega}), \tag{11}$$

where the next state value is estimated by a weighted sum of the value of continuing the current option and the maximum value in the next state, with the weights determined by the termination value.

### 4.5 Model Training

As illustrated in Algorithm 1, in MTAMCR, the CRS needs to generate a complete chain of choices before interacting with the user and obtaining feedback. However, during the training stage, we can receive real feedback for each choice $a_t$ and reward $r_t$ immediately from users at each timestep $t$, and accurately infer the next state $s_{t+1}$, which speeds up the convergence of training. Each option $\omega$ has its own experience replay buffer, denoted by $\mathcal{D}\omega$, and the experience $d_t = (s_t, a_t, r_t, s_{t+1}, \mathcal{A}_{t+1}, \omega_T)$ is stored in the replay buffer $\mathcal{D}\omega_T$. We sample

a mini-batch of experiences from each $\mathcal{D}_{\omega_T}$ after every turn and optimize the parameters of the whole network.

In this work, we implement the long policy over options $\pi_\Omega$ and short intra-option policies $\pi_\omega$ to sample the probability distribution of the discrete action space with the Q-value function as follows:

$$\pi_\Omega(s_t) = \arg\max_{\omega} Q_\Omega(s_t, \omega), \tag{12}$$

$$\pi_{\omega_T}(a_t \mid s_t) = \frac{\exp(Q_U(s_t, \omega_T, a_t))}{\sum_{a \in \mathcal{A}_t} \exp(Q_U(s_t, \omega_T, a))}, \tag{13}$$

where both $\pi_\Omega$ and $\pi_\omega$ freeze the parameters for the stable training optimization and only need a learnable $Q_U$ function to represent $\pi_\Omega$, $\pi_\omega$ and $Q_\Omega$.We devise a dueling deep Q-network [16] to estimate the expected value of the triplet $(s_t, \omega_t, a_t)$ at timestep $t$ and the Q-value of the tuple can be estimated by:

$$Q_U(s_t, \omega_T, a_t; \theta_Q, \theta_S) = f_{\theta_V}(f_{\theta_S}(s_t)) + f_{\theta_A}(f_{\theta_S}(s_t), \omega_t, a_t), \tag{14}$$

where $\theta_V$ and $\theta_A$ represent the parameters of the value and advantage functions, respectively, simplified as $\theta_Q = \{\theta_V, \theta_A\}$. The target value $y_t$ of $Q_U$ during training can be obtained as Eq. (10), and we can optimize the parameters of the Q network by minimizing the MSE loss as follows:

$$\mathcal{L}(\theta_Q, \theta_S) = \mathbb{E}_{d_t \sim \mathcal{D}}[(y_t - Q_U(s_t, \omega_T, a_t; \theta_Q, \theta_S))^2]. \tag{15}$$

Based on the Termination Gradient Theorem proposed by [1], the parameters $\theta_\beta$ of termination network $\beta_\omega$ can be optimized by increasing the gradient or minimizing the loss:

$$\mathcal{L}(\theta_\beta) = \mathbb{E}_{d_t \sim \mathcal{D}}[\beta_\omega(s_{t+1}; \theta_\beta)(Q_\Omega(s_{t+1}, \omega_T) - V(s_{t+1}))], \tag{16}$$

where $V(s_{t+1}) = f_{\theta_V}(f_{\theta_S}(s_{t+1})$ is the estimated value of the next state. Intuitively, when the value of selecting the current option exceeds the estimated value, the probability of terminating the option is reduced.

For the feedback prediction function, we simply implement it by a multilayer perceptron (MLP). With the ground-truth state and reward, we can optimize it by minimizing the following loss to minimize the squared difference between the predicted feedback for action $a_t$ and the ground-truth acceptance or rejection by the user:

$$\mathcal{L}(\theta_\mathcal{T}) = \mathbb{E}_{d_t \sim \mathcal{D}}[(\mathbb{1}(a_t \text{ is accepted}) - \mathcal{T}_{\omega_t}(s_t, a_t; \theta_\mathcal{T}))^2], \tag{17}$$

where $\mathbb{1}(\cdot)$ takes the value of 1 when user accepts $a_t$, and 0 otherwise.

## 5 Experiments

To demonstrate the superiority of CoCHPL[1] in the MTAMCR scenario, we conduct experiments aimed at validating the following three research questions (RQs):

- **RQ1:** With the comparison of state-of-the-art methods, how much does our model CoCHPL improve in overall performance?
- **RQ2:** How do different components contribute to the performance?
- **RQ3:** Does our model CoCHPL improve the diversity and dependency between attributes in each turn?

### 5.1 Experimental Settings

**Datasets.** To evaluate our proposed framework, we employ four public benchmark datasets that are commonly used for conversational recommendation tasks involving multiple choice questions, namely: LastFM[2], Yelp[3], Amazon-Book[4] and MovieLens[5]. The statistics of datasets are shown in Table 1.

- **LastFM**: LastFM is a popular online platform that focuses on music recommendations and social networking. The original attributes in LastFM were treated as attribute instances, and a clustering method was utilized to identify 34 attribute types.
- **Yelp**: Yelp serves for business recommendations. In the context of Yelp, a two-layer taxonomy was created by [6], comprising 29 attribute types and 590 attribute instances. The attribute types in the taxonomy represent different categories or aspects of businesses, such as price range, ambiance, service quality, and more.
- **Amazon-Book**: The dataset is widely used for book recommendations. Users and items with a minimum of 10 interaction records are retained. Entities and relations in the knowledge graph are considered as attribute instances and attribute types, respectively.
- **MovieLens**: MovieLens is a movie rating dataset. It contains user ratings for movies, which can be utilized to build recommendation models and understand user preferences. Interactions with ratings above three are retained, and the knowledge graph entities and relations are considered as attribute instances and attribute types.

[1] Our code is publicly available at: https://github.com/AlexFanw/CoCHPL
[2] https://grouplens.org/datasets/hetrec-2011/
[3] https://www.yelp.com/dataset/
[4] http://jmcauley.ucsd.edu/data/amazon
[5] https://grouplens.org/datasets/movielens

**Table 1.** Statistics of four benchmark datasets.

| | **LastFM** | **Yelp** | **Amazon-Book** | **MovieLens** |
|---|---|---|---|---|
| #Users | 1,801 | 27,675 | 30,291 | 20,892 |
| #Items | 7,432 | 70,311 | 17,739 | 16,482 |
| #Attributes | 8,438 | 590 | 988 | 1,498 |
| #Types | 34 | 29 | 40 | 24 |
| #Entities | 17,671 | 98,576 | 49,018 | 38,872 |
| #Relations | 4 | 3 | 2 | 2 |
| #Interactions | 76,693 | 1,368,606 | 478,099 | 454,011 |
| #Triplets | 228,217 | 2,533,827 | 565,068 | 380,016 |

**Baselines.** To evaluate and demonstrate the performance of our model, we choose the following eight baselines for comparison.

- **Abs Greedy** [3] only recommends items in each turn until the user accepts the recommendation or the total turn number exceeds the maximum limit.
- **Max Entropy** [6] asks or recommends the top-scored instances with the maximum entropy with a certain probability.
- **CRM** [12] introduces a policy that selects the action and terminates the conversation when the recommendation is rejected.
- **EAR** [6] proposes a three-stage pipeline called Estimation–Action-Reflection for selecting actions and optimizing the model.
- **SCPR** [8] employs graph-based path reasoning to retrieve actions and graph embedding to represent user preferences dynamically.
- **UNICORN** [4] offers a comprehensive approach for selecting optimal actions, integrating both the conversation and recommendation components.
- **MCMIPL** [19] proposes single-type attributes based on multiple-choice question scenarios and uses a multi-interest extractor to capture user preferences.
- **DAHCR** [21] employ one director to select the action type and one actor to choose the asked attribute or recommended item.

To ensure a fair comparison in the MTAMCR scenario, we made two kinds of modifications to UNICORN, MCMIPL, and DAHCR:

- **Single-Type attributes (-S)**: Also known as attribue type-based multiple choice [19]. In each round, CRS selected one attribute type and then chose multiple attribute instances within that type.
- **Multi-Type attributes (-M)**: In each round, we selected the top-K attribute instances with the highest Q-values, which could be from different or the same attribute types.

**Table 2.** Overall performance evaluated by `SR@15`, `AT`, and `hDCG` across four datasets.

| Model | LastFM | | | Yelp | | | Amazon-Book | | | MovieLens | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | SR@15 | AT | hDCG | SR@15 | AT | hDCG | SR@15 | AT | hDCG | SR@15 | AT | hDCG |
| Abs Greedy[3] | 0.635 | 8.66 | 0.267 | 0.189 | 13.43 | 0.089 | 0.193 | 13.90 | 0.087 | 0.724 | 5.24 | 0.465 |
| Max Entropy[6] | 0.669 | 9.33 | 0.269 | 0.398 | 13.42 | 0.121 | 0.382 | 12.43 | 0.149 | 0.655 | 6.87 | 0.412 |
| CRM[12] | 0.580 | 10.79 | 0.224 | 0.177 | 13.69 | 0.070 | 0.354 | 12.81 | 0.127 | 0.626 | 7.43 | 0.433 |
| EAR[6] | 0.595 | 10.51 | 0.230 | 0.182 | 13.63 | 0.079 | 0.411 | 11.62 | 0.153 | 0.749 | 6.28 | 0.459 |
| SCPR[8] | 0.709 | 8.43 | 0.317 | 0.489 | 12.62 | 0.159 | 0.432 | 11.03 | 0.148 | 0.823 | 4.94 | 0.514 |
| UNICORN[4] | 0.788 | 7.58 | 0.349 | 0.520 | 11.31 | 0.203 | 0.498 | 10.42 | 0.179 | 0.863 | 4.62 | 0.523 |
| DAHCR[21] | 0.925 | 6.31 | 0.431 | 0.626 | 11.02 | 0.192 | 0.522 | 10.14 | 0.188 | 0.875 | 4.34 | 0.535 |
| UNICORN-S | 0.884 | 6.07 | 0.403 | 0.593 | 10.57 | 0.212 | 0.538 | 10.01 | 0.253 | 0.883 | 4.05 | 0.585 |
| MCMIPL-S | 0.942 | 5.44 | 0.431 | 0.631 | 10.25 | 0.225 | 0.556 | 9.57 | 0.264 | 0.879 | 3.92 | 0.581 |
| DAHCR-S | 0.933 | 5.67 | 0.429 | 0.668 | 10.15 | 0.232 | 0.572 | 9.45 | 0.272 | 0.895 | 3.89 | 0.603 |
| UNICORN-M | 0.892 | 5.96 | 0.426 | 0.634 | 10.35 | 0.233 | 0.584 | 9.69 | 0.261 | 0.887 | 4.10 | 0.587 |
| MCMIPL-M | 0.973 | 4.59 | 0.466 | 0.675 | 10.11 | 0.256 | 0.667 | 8.98 | 0.293 | 0.891 | 3.68 | 0.609 |
| DAHCR-M | 0.965 | 4.61 | 0.451 | 0.681 | 10.23 | 0.241 | 0.689 | 8.45 | 0.301 | 0.874 | 3.98 | 0.593 |
| **CoCHPL** | **0.991** | **3.75** | **0.493** | **0.846** | **9.36** | **0.267** | **0.775** | **7.61** | **0.330** | **0.908** | **2.85** | **0.628** |

**Evaluation Metrics.** We evaluate the performance of our MCR recommendation system using three key metrics: (1) Success rate (**SR@t**): Measures the cumulative ratio of successful recommendations within a maximum of $t$ turns. (2) Average turn (**AT**): Evaluates the average number of terminated conversation turns for all sessions. Lower values indicate higher efficiency in the recommendation process. (3) **hDCG@(T, K)** [4]: Assesses the ranking performance of recommended items by considering the target item's rank in the recommendation list. It extends the normalized discounted cumulative gain metric.

**Training Details.** Based on [4], we set the maximum turns $T$ that the user can tolerate to 15, and the agent can recommend up to $k_v = 10$ items or ask $k_p = 3$ attribute-related questions at every turn. We pre-train the graph nodes with TransE [2], train each dataset for $10,000$ episodes online, and the batch size for experience replay is 64. Adam optimizer with the learning rate $1e-4$. Discount factor $\gamma$ is set to be 0.999. For our method CoCHPL, the reward settings were as follows: $r_{ask\_acc} = 1e-2$, $r_{ask\_rej} = -1e-4$, $r_{rec\_rej} = -1e-4$, and $r_{rec\_suc} = 1$. We found that the effect of retrieving accepted attributes was minimal in MovieLens, and then the reward for $r_{ask\_acc}$ was adjusted to $1e-5$ for it. The rewards for all state-of-the-art baselines remain unchanged as [4].

### 5.2 Experimental Results

**Overall Performance (RQ1).** Table 2 presents the experimental results of CoCHPL and all baselines. CoCHPL outperforms the eight baselines and their variants in terms of SR@15, AT, and hDCG. The baselines trained on the binary yes/no question setting generally have lower efficiency and require more

**Table 3.** Ablation Study evaluated across Yelp and Amazon-Book datasets.

| | Yelp | | | Amazon-Book | | |
|---|---|---|---|---|---|---|
| | SR@15 | AT | hDCG | SR@15 | AT | hDCG |
| **CoCHPL** | **0.846** | **9.36** | **0.267** | **0.775** | **7.61** | **0.330** |
| w/o long policy over options | 0.693 | 10.21 | 0.225 | 0.649 | 9.12 | 0.278 |
| w/o short intra-option policy (ask) | 0.655 | 10.29 | 0.221 | 0.612 | 9.54 | 0.270 |
| w/o short intra-option policy (rec) | 0.756 | 10.05 | 0.245 | 0.685 | 8.99 | 0.285 |
| w/o termination function | 0.798 | 9.67 | 0.253 | 0.744 | 8.34 | 0.302 |
| w/o feedback prediction function | 0.782 | 9.95 | 0.248 | 0.732 | 8.12 | 0.311 |

turns for successful recommendations, as illustrated in Figure 1(a). Under the single-type attributes setting, UNICORN-S, MCMIPL-S, and DAHCR-S show limited improvement in performance compared to their original versions. Under the multi-type attributes setting, UNICORN-M, MCMIPL-M, and DAHCR-M demonstrate better success rates and shorter average turns, which indicates that our MTAMCR setting is more effective in capturing user preferences. With the chain-of-choice generation, CoCHPL not only improves success rates and recommendation efficiency across four benchmark datasets but also significantly reduces the number of turns required for successful recommendations. Notably, on the Yelp and Amazon-Book datasets, where the retrieval of preferred attributes is crucial, CoCHPL achieves relative improvements of 18.3%, 8.6%, and 6.9% in SR@15, AT, and hDCG, compared to the state-of-the-art methods.

**Ablation Study (RQ2).** We conducted an ablation study on the Yelp and Amazon-Book datasets to assess the impact of different components on system performance as follows: (1) `w/o policy over options` removes the long policy over options, and the agent randomly selects options (action types) in each turn. (2) `w/o intra-option policy (ask)` removes the short intra-option policy of asking. Instead, the agent chose the top-$K$ attribute instances with the highest Q-value at every turn. (3) `w/o intra-option policy (recommend)` removes the short intra-option policy of recommending, and the agent selected the top-$K$ items with the highest Q-value at each turn. (4) `w/o termination function` removes the termination function, causing the system to ask for or recommend the maximum number of instances in each turn without appropriate termination. (5) `w/o feedback prediction function` removes the user feedback prediction function. The CRS randomly predicts whether the user would accept or reject the previous choice. From Table 3, we can observe that `short intra-option policy (ask)` and `long policy over options` play crucial roles in performance improvement, which suggests that learning how to select options and effectively generate multiple choice questions about attributes is essential in the MTAMCR scenario. The impact of the termination and feedback prediction function highlights the importance of learning appropriate termination condi-

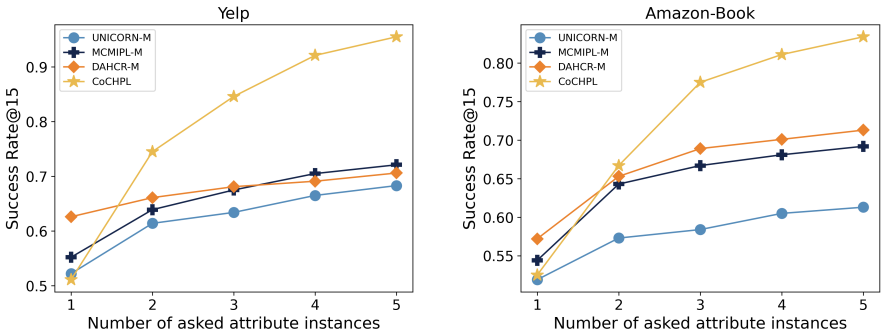
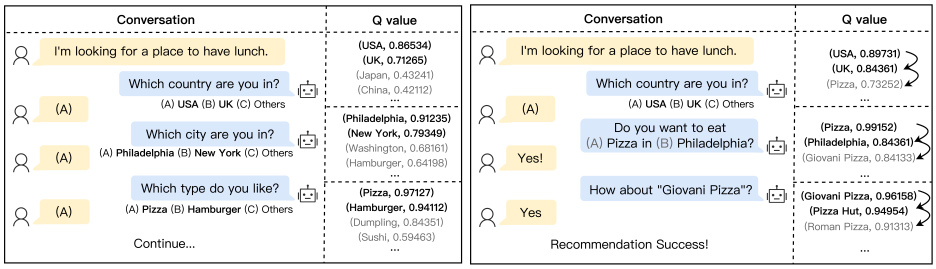



**Fig. 4.** Performance comparisons of different asked attribute instances numbers in Yelp and Amazon-Book datasets.

tions across options and underscores the necessity of simulating user behavior during the chain-of-choice generation.

**Effect of Attribute Instances Number (RQ3).** To demonstrate the enhanced reasoning ability of our model, we conducted tests on the Yelp and Amazon-Book datasets, focusing on different asked attribute instances as shown in Figure 4. In the MTAMCR scenario, we compared our approach, CoCHPL, with three baselines: UNICRON-M, MCMIPL-M, and DAHCR-M. In each round, the CRS could ask for 1 to 5 attribute instances, and the performance is measured using the success rate in 15 turns. We observed that although the performance of the three baselines improved as the number of attribute instances increased, the degree of improvement gradually diminished and approached zero. This indicates that these methods select many ineffective attribute instances in each round, which causes inefficiently exploiting user preferences. In contrast, our CoCHPL method exhibited nearly linear growth on both datasets, demonstrating that the Chain-of-Choice Policy proposed by CoCHPL enhances the logical coherence among questions in each turn and reduces the similarity and redundancy among attributes. The results suggest that CoCHPL effectively leverages the power of chain-based reasoning and improves the overall performance.

**Case Study (RQ3).** To further demonstrate how our method enhances the diversity and dependency of attributes, we present a conversation case as shown in Figure 5. The state-of-the-art method DAHCR computes and ranks the Q-values for all candidate attribute instances and then selects the top-2 attribute instances as choices for the question. However, in the DAHCR method, attribute instances with similar Q-values in the same state are likely to belong to the same type because their embeddings are similar. In contrast, our method predicts the user's feedback and the next state after choosing a choice. The agent recalculates the Q-values on the next state and selects the top-1 attribute as the new choice. The




(a) DAHCR-M (b) CoCHPL

**Fig. 5.** Conversations generated by DAHCR-M and CoCHPL, and the process of choosing choices in each turn. By utilizing prediction and inference stages, CoCHPL is capable of generating a more coherent and logical sequence of choices.

chain-of-choice process effectively expands the logical relevance among choices and improves the success rate in the multi-type attributes scenario.

## 6 Conclusion

In this paper, we propose a more realistic CRS setting called Multi-Type-Attribute Multi-round Conversational Recommendation (MTAMCR). To optimize the questioning efficiency, we develop a novel framework, namely Chain-of-Choice Hierarchical Policy Learning (CoCHPL), for the MTAMCR scenario. Specifically, by formulating MTAMCR as a hierarchical RL task, we employed a long-term policy over action options to decide when to ask or recommend and two short-term intra-option policies to sequentially generate the optimal multi-type attribute and item chains with the assistance of the learnable termination and feedback prediction functions. Experimental results demonstrate that CoCHPL significantly outperforms state-of-the-art CRS methods in terms of both the reasoning ability in chains of choice and the success rate of recommendations.

## References

1. Bacon, P.L., Harb, J., Precup, D.: The option-critic architecture. In: Proceedings of the AAAI conference on artificial intelligence. vol. 31 (2017)
2. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. NeurIPS **26** (2013)
3. Christakopoulou, K., Radlinski, F., Hofmann, K.: Towards conversational recommender systems. In: SIGKDD. pp. 815–824 (2016)
4. Deng, Y., Li, Y., Sun, F., Ding, B., Lam, W.: Unified conversational recommendation policy learning via graph-based reinforcement learning. In: SIGIR. pp. 1431–1441 (2021)
5. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)

6. Lei, W., He, X., Miao, Y., Wu, Q., Hong, R., Kan, M.Y., Chua, T.S.: Estimation-action-reflection: Towards deep interaction between conversational and recommender systems. In: WSDM. pp. 304–312 (2020)
7. Lei, W., He, X., de Rijke, M., Chua, T.S.: Conversational recommendation: Formulation, methods, and evaluation. In: SIGIR. pp. 2425–2428 (2020)
8. Lei, W., Zhang, G., He, X., Miao, Y., Wang, X., Chen, L., Chua, T.S.: Interactive path reasoning on graph for conversational recommendation. In: SIGKDD. pp. 2073–2083 (2020)
9. Lin, A., Zhu, Z., Wang, J., Caverlee, J.: Enhancing user personalization in conversational recommenders. arXiv preprint arXiv:2302.06656 (2023)
10. Precup, D.: Temporal abstraction in reinforcement learning. University of Massachusetts Amherst (2000)
11. Priyogi, B.: Preference elicitation strategy for conversational recommender system. In: WSDM. pp. 824–825 (2019)
12. Sun, Y., Zhang, Y.: Conversational recommender system. In: SIGIR (2018)
13. Sutton, R.S., McAllester, D., Singh, S., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. NeurIPS **12** (1999)
14. Sutton, R.S., Precup, D., Singh, S.: Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. Artificial intelligence **112**(1-2), 181–211 (1999)
15. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. NeurIPS **30** (2017)
16. Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., Freitas, N.: Dueling network architectures for deep reinforcement learning. In: ICML. pp. 1995–2003. PMLR (2016)
17. Xie, Z., Yu, T., Zhao, C., Li, S.: Comparison-based conversational recommender system with relative bandit feedback. In: SIGIR. pp. 1400–1409 (2021)
18. Xu, K., Yang, J., Xu, J., Gao, S., Guo, J., Wen, J.R.: Adapting user preference to online feedback in multi-round conversational recommendation. In: WSDM. pp. 364–372 (2021)
19. Zhang, Y., Wu, L., Shen, Q., Pang, Y., Wei, Z., Xu, F., Long, B., Pei, J.: Multiple choice questions based multi-interest policy learning for conversational recommendation. In: WWW. pp. 2153–2162 (2022)
20. Zhang, Y., Chen, X., Ai, Q., Yang, L., Croft, W.B.: Towards conversational search and recommendation: System ask, user respond. In: CIKM. pp. 177–186 (2018)
21. Zhao, S., Wei, W., Liu, Y., Wang, Z., Li, W., Mao, X.L., Zhu, S., Yang, M., Wen, Z.: Towards hierarchical policy learning for conversational recommendation with hypergraph-based reinforcement learning (2023)