# User Simulator-Guided Multi-Turn Preference Optimization for Reasoning LLM-based Conversational Recommendation

**Anonymous ACL submission**


## Abstract

Conversational Recommender Systems (CRSs) leverage natural language interactions to provide personalized recommendations. However, sparse dialogue histories and single-turn recommendation paradigms often hinder accurate modeling of complex user preferences. Recent studies have introduced Large Language Model (LLM)-based user simulators to enable simulated multi-turn interactions with natural language feedback. Nevertheless, since simulators cannot access ground-truth user preferences during inference, their feedback may deviate from real user interests, leading to error accumulation and degraded recommender generalization. To address these challenges, we propose SMTPO, a user simulator-guided multi-turn preference optimization framework for conversational recommendation. SMTPO transforms potentially biased simulator feedback into policy-level optimization signals instead of merely using it for candidate filtering. It trains a user simulator to provide informative feedback and optimizes a reasoning LLM-based recommender via supervised fine-tuning and multi-turn reinforcement learning, where fine-grained rewards progressively align recommendation policies with true user preferences under potentially biased feedback. Extensive experiments on public datasets demonstrate the effectiveness and transferability of SMTPO across different settings. Our code is available at https://anonymous.4open.science/r/SMTPO_3602-4CFA.


## 1 Introduction

Conversational Recommender Systems (CRSs) aim to provide personalized recommendation to users via natural language interactions. The key challenge of CRS lies in the inherently brief nature of typical conversations, easily leading to insufficient historical context in modeling user preferences. As such, prior work often leverages external information (e.g., knowledge graphs (KGs) (Chen et al., 2019; Zhou et al., 2020) or reviews (Zheng et al., 2024b)) to enhance user preference understanding. Recently, Large Language Models (LLMs) have been integrated into CRS for their strong semantic understanding capabilities. Despite success, LLMs often struggle to effectively incorporate collaborative filtering (CF) (Wu et al., 2024; Zhu et al., 2024; Zheng et al., 2024a), a cornerstone of traditional recommendation systems. Accordingly, sequential research on LLM-driven CRS has increasingly focused on hybrid approaches to effectively overcome such issues, for instance, retrieval-reranking paradigm to retrieve candidates for LLM-based reranking (Yang and Chen, 2024), or retrieval-augmented generation to seamlessly integrate LLMs with CF (Zhu et al., 2025). Nonetheless, user preferences are often diverse and complex, so relying solely on single-turn interaction may limit the accuracy of CRSs (Wang et al., 2025), while excessive engagement may severely degrade the overall conversational experience.

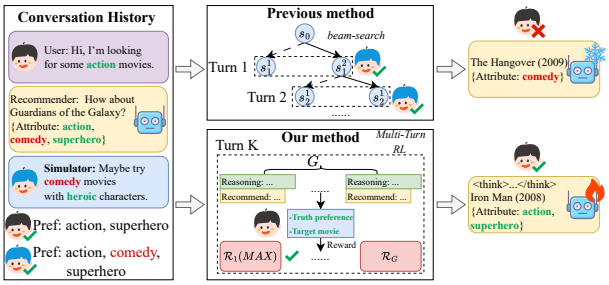



Figure 1: Previous methods froze recommender parameters and relied on beam search with simulator-filtered results, making it sensitive to feedback bias. Our method trains the SFT-initialized recommender via RL to achieve robust and accurate recommendation.

To this end, recent works (Feng et al., 2025; Yoon et al., 2024; Wang et al., 2023) have explored leveraging LLM-based user simulators to enhance CRSs' understanding of complex user preferences and improve recommendation performance. These methods typically comprise two components, i.e.,

a user simulator modeling user preferences and a recommender generating context-aware recommendations based on dynamically generated user feedback. For instance, ECPO (Feng et al., 2025) introduces expectation confirmation to identify unsatisfactory turns and construct turn-level preference pairs for response alignment, GRSU (Wang et al., 2025) employs a frozen LLM-based recommender to produce candidates via beam search and then uses a generative reward model-based simulator for optimal selection. Despite their effectiveness, these methods still largely rely on simulator-generated feedback or judgments, which may deviate from true user interests and thus be detrimental to the generalization of CRSs. As shown in Figure 1, the simulator incorrectly interprets the user's preference as "comedy," thereby providing misleading feedback that biases the recommender's filtering decisions. As such feedback is repeatedly used in subsequent interaction rounds, the beam-search process may gradually accumulate errors and deviate from the user's actual preferences, ultimately degrading recommender generalization.

These observations highlight two key challenges for simulator-guided CRSs. (1) Simulator feedback is inevitably imperfect, as ground-truth user preferences are unavailable during inference; thus, improving simulator realism alone cannot fully eliminate feedback bias. (2) Recommender should not passively follow such feedback, but learn to extract reliable preference signals from noisy interactions and progressively refine its preference estimation across turns. This calls for moving beyond post-hoc candidate filtering toward policy-level optimization under potentially biased simulator feedback.

To address these challenges, drawing on the success of practices of LLM-based agents (Dong et al., 2024; Huang et al., 2024; Zhang et al., 2026) as well as the strong multi-step reasoning capabilities of Reasoning LLMs (Fang et al., 2025; Zhao et al., 2025), we propose SMTPO (**S**imulator **M**ulti-**T**urn **P**reference **O**ptimization), a simulator-guided CRS framework based on Reasoning LLM. The key idea of SMTPO is to transform simulator-generated feedback from a post-hoc filtering signal into a policy-level optimization signal. Specifically, SMTPO first trains a user simulator to generate informative feedback, infer target-related attributes, helping the recommender make accurate recommendations. Then, instead of keeping the recommender frozen, SMTPO optimizes a reasoning LLM-based recommender through a two-stage process: supervised fine-tuning teaches the model task-specific reasoning and ranking patterns, while multi-turn reinforcement learning (RL) further aligns its preference inference and ranking decisions with true user preferences under potentially biased feedback. In addition, a dual semantic-collaborative retriever dynamically updates the candidate set, ensuring that the reasoning LLM operates within the valid global item space. Our contributions are summarized as follows:

- We propose SMTPO, a simulator-guided multi-turn preference optimization framework for CRSs. Unlike prior methods that mainly use simulator feedback for evaluation or candidate filtering, SMTPO optimizes the recommender policy itself under imperfect simulator feedback.
- We design a robust two-stage recommender optimization strategy. Fine-grained SFT-RL optimization converts potentially biased simulator feedback into policy-level signals, while the simulator and retriever jointly support dynamic, preference-aware candidate generation.
- Extensive experiments further demonstrate that our method significantly outperforms existing baselines on multiple datasets, verifying its effectiveness and superiority.

## 2 Methodology

In this paper, we propose a multi-turn preference optimization framework for conversational recommendation, named **SMTPO**, which is composed of three core modules: **user simulator**, **retriever** and Reasoning LLM-based **recommender**. The overall framework is illustrated in Figure 2.

### 2.1 Preliminaries

We consider a conversational recommendation task, where the goal is to recommend items that match user preference after multi-turn interaction. At each turn $t$, the user simulator generates feedback $F_t$ without access to ground-truth preference labels. The feedback is based on the conversation history $D$, the mentioned item-attribute pairs $I_D = (i_1, a_1), \ldots, (i_n, a_n)$, where $i$ represents an item and $a$ denotes the corresponding attribute, and the previous recommendation list $R_{t-1}$. To improve efficiency, a retriever first constructs a candidate set $I^t_{\text{cand}}$ by combining semantic information from $(D, F_t, P_{t-1})$ and collaborative signals from $(I_D, R_{t-1})$ using a pre-trained knowledge graph

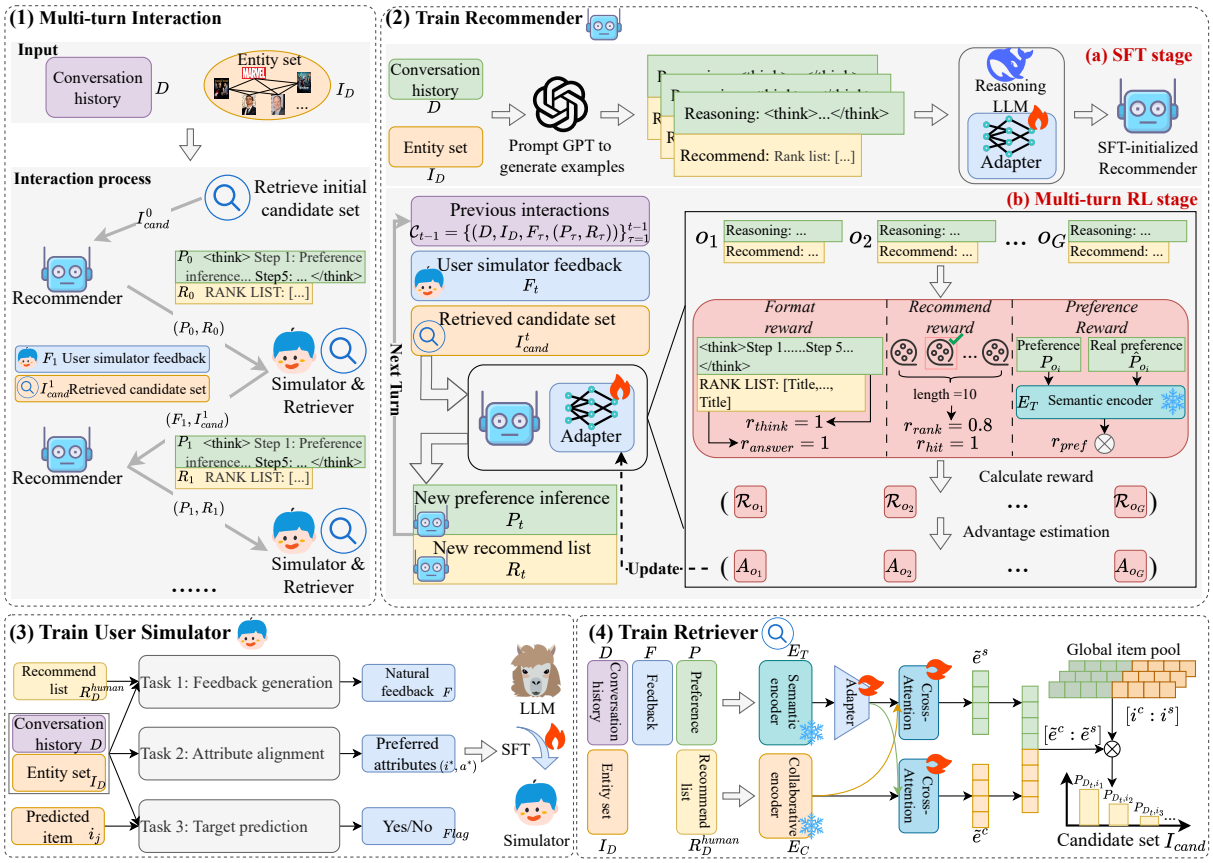



Figure 2: Overview of SMTPO training and multi-turn interaction: (1) The multi-turn interaction process. (2) The recommender is trained first with SFT and then with multi-turn RL. (3) The simulator is trained via multi-task SFT. (4) The retriever is obtained using collaborative–semantic dual-view modeling.

$\mathcal{G}$. The LLM-based recommender then infers user preference $P_t$ and reranks $I^t_{\text{cand}}$ to produce the recommendation list $R_t$. The interaction history up to turn $t$ is defined as follows:

$$\mathcal{C}_t = \big((P_0, R_0), \{F_\tau, (P_\tau, R_\tau)\}_{\tau=1}^{t}\big) \qquad (1)$$

The process at each turn is formulated as:

$$\begin{cases} F_t = \text{Simulator}(D, I_D, R_{t-1}) \\ I^t_{\text{cand}} = \text{Retriever}(D, I_D, P_{t-1}, R_{t-1}, F_t; \mathcal{G}) \\ (P_t, R_t) = \text{Recommender}(D, I_D, \mathcal{C}_{t-1}, F_t, I^t_{\text{cand}}) \end{cases} \qquad (2)$$

where $t \in \{1, \ldots, T\}$ and $R_0$ is initialized by the retriever. The interaction continues until turn $T$ or stops early once the target item $i^*$ is recommended.

## 2.2 Multi-Turn Interaction

To ensure stable multi-turn training, we first train the simulator and retriever independently, followed by the recommender. The simulator is optimized via multi-task supervised fine-tuning (SFT) to generate high-quality user feedback; the retriever is trained with collaborative-semantic dual-view modeling to recall candidate sets; and the recommender adopts a two-stage training strategy to progressively improve preference reasoning and recommendation performance over multiple turns. Since the original data only contains dialogues, we further use GPT-3.5-Turbo (Shafik, 2024) to construct training data, including inferred user preferences and simulated user feedback. Detailed data construction procedures and training prompts are provided in Appendix B.

The multi-turn interaction process of SMTPO is shown in Figure 2. In the first turn, the retriever obtains an initial candidate set $I^0_{\text{cand}}$ from the conversation history $D$ and entity set $I_D$, and the recommender generates initial preference reasoning $P_0$ and recommendation list $R_0$. In each subsequent turn $t$, as shown in Eq. 2, the simulator provides feedback $F_t$, the retriever updates the candidate set $I^t_{\text{cand}}$, and the recommender updates $P_t$ and $R_t$.

## 2.3 User Simulator: Generate Feedback

The user simulator generates feedback for multi-turn preference elicitation. Since ground-truth preferences are unavailable during inference, we train it with multi-task SFT to produce informative feed-

back that better reflects real user preference.

#### 2.3.1 Feedback Generation Task

Considering that the user simulator should provide specific and informative feedback, we design a feedback generation task. Given the conversation history $D$, the mentioned entity set $I_D$, and a human-constructed recommendation list $R_D^{\text{human}}$, the simulator generates user feedback. To reflect realistic multi-turn interactions, we construct $R_D^{\text{human}}$ using four strategies: hard negatives with/without the target item and simple negatives with/without the target item, denoted as $H^+$, $H^-$, $S^+$, and $S^-$. Negative sample difficulty is determined by attribute overlap with the target item. Details can be found in Algorithm 1. The task is formally defined as follows:

$$F = LLM_{simulator}(D, I_D, R_D^{\text{human}}) \quad (3)$$

where $R_D^{\text{human}} \in \{H^+, H^-, S^+, S^-\}$.

#### 2.3.2 Attribute Alignment Task

To ensure that the simulator generates high-quality feedback, we require the simulator to resemble real users not only in style and semantics but also in alignment with the target item at the attribute level for finer-grained alignment with user preferences. Thus, we design an attribute alignment task, where the simulator predicts the target item $i^*$ and its attributes $a^*$ based on the conversation history $D$ and the entity set $I_D$:

$$(i^*, a^*) = LLM_{simulator}(D, I_D) \quad (4)$$

#### 2.3.3 Target Prediction Task

During multi-turn interactions, the recommendation list generated by the recommender may contain noise. Specifically, some candidate items may have semantics inconsistent with the user's true preferences. The simulator requires strong judgment to filter out these irrelevant items, so we designed a target prediction task. Let the simulator determine whether the current item $i_j$ (where $i_j \in H^+ \cup H^- \cup S^+ \cup S^-$) is the target item of the user, based on $D$ and $I_D$. The simulator should return a variable $Flag$ ("Yes" or "No") as its response. The formal definition is as follows:

$$Flag = LLM_{simulator}(i_j | D, I_D) \quad (5)$$

**Model training.** We adopt LoRA (Hu et al., 2022) to perform multi-task SFT on the LLM to obtain the user simulator. During training, for each task instance $(\mathcal{I}, o) \in Inst$, the model updates the parameters $\Theta^u$ by maximizing the conditional probability of the target output sequence. The loss function is defined as follows:

$$\mathcal{L}_1 = -\sum_{(\mathcal{I},o)\in Inst} \sum_{i=1}^{|o|} \log P_{\Theta^u}(o_i | o_{<i}, \mathcal{I}) \quad (6)$$

Here, $\mathcal{I}$ and $o$ denote the input and output, respectively, and $Inst$ is the set of all training instances.

### 2.4 Retriever: Recall Candidate Set

To avoid the LLM-based recommender generating items outside the global item space, we adopt a retrieval–reranking paradigm and design a dual-view retriever. It encodes each dialogue from semantic and collaborative perspectives to retrieve a candidate set from the global item pool.

#### 2.4.1 Feature Embedding

**Semantic Embedding.** On the semantic side, we use a pre-trained language model (PLM) as the semantic encoder $E_T$, which encodes the conversation history $D$, user feedback $F$, and the recommender's textual preference inference $P$ to obtain semantic representation $e^s$:

$$e^s = E_T(D, F, P) \quad (7)$$

**Collaborative Embedding.** Semantic encoder mainly capture dialogue semantics but may overlook structural relations between movies and attributes. Therefore, we construct a knowledge graph from WikiMKG (Qiu et al., 2024), with movies and attributes as nodes and their relations as edges. We pretrain a collaborative encoder $E_C$ with BPR loss, where related movie and attribute pairs are positive samples and unrelated pairs are negatives. After pretraining, $E_C$ is frozen to encode $I_D$ and the human-constructed recommendation list $R_D^{human}$, and these embeddings are mean-pooled to obtain the collaborative representation $e^c$:

$$e^c = AvgPool(E_C(I_D, R_D^{human})) \quad (8)$$

#### 2.4.2 Dual-view Modeling

After obtaining the initial features, in order to better integrate the conversation representations of the two views, we adopt a cross-attention mechanism for feature fusion. Since the semantic dimension $Dim_s$ differs from the collaborative dimension $Dim_c$, spatial alignment is required first. We use an adapter to map semantic information

$e^s$ to the collaborative space. This process can be formally defined as follows:

$$\hat{e}^s = W_2(W_1 e^s + b_1) + b_2 \tag{9}$$

Where $W_1 \in R^{Dim_s \times \frac{Dim_s}{2}}$, $W_2 \in R^{\frac{Dim_s}{2} \times Dim_c}$, $b_1 \in R^{\frac{Dim_s}{2} \times 1}$, $b_2 \in R^{Dim_c \times 1}$ are weight matrices and bias of the adapter.

After completing spatial alignment, feature fusion is performed based on cross-attention. Taking the collaborative side as an example, we set $\hat{e}^s$ as the query vector and $e^c$ as the key and value vector. Let $Q_s = \hat{e}^s W^Q$, $K_c = e^c W^K$,$V_c = e^c W^V$, where $W^Q$,$W^K$,$W^V \in R^{Dim_c \times Dim_c}$, the collaborative feature that integrates semantic side information can be represented as:

$$\tilde{e}^c = Softmax(\frac{Q_s K_c^T}{\sqrt{d_k}})V_c \tag{10}$$

The same approach can be used to obtain semantic features that integrate collaborative side information $\tilde{e}^s$. By concatenating these two fusion vectors, the feature representation of conversation $D$ can be obtained $e = [\tilde{e}^c : \tilde{e}^s]$. Similarly, we concatenate the text features $i^s$ and collaborative features $i^c$ of the item as the fused features on the item side. We then retrieve the recommended candidate set by calculating feature similarity. For the given conversation $D$ and item $i$, the recall score is calculated as follows:

$$P_{D,i} = [\tilde{e}^c : \tilde{e}^s]^T [i^c : i^s] \tag{11}$$

where $i^s$ and $i^c$ are encoded by the text encoder $E_T$ and graph encoder $E_C$, respectively. Finally, we rank and return the top-k items by recall score as a candidate item set for conversation $D_t$ in the t-th turn of interaction.

**Model training.** Based on the fused semantic and collaborative representations defined above, the retriever is optimized to assign higher recall scores to the target item than to irrelevant items. During this stage, the semantic encoder $E_T$ and the collaborative encoder $E_C$ are kept frozen, and only the parameters of the adapter and fusion modules in the retriever are updated. For each conversation $D$, its ground-truth target item $i^*$ is treated as the positive sample, while negative items are randomly sampled from the global item pool. The retriever is trained using the InfoNCE loss:

$$\mathcal{L}_2 = -\log \frac{e^{P_{D,i^*}}}{e^{P_{D,i^*}} + \sum_{k=1}^{N} e^{P_{D,i_k^{neg}}}} \tag{12}$$

where $i_k^{neg}$ denotes the $k$-th negative item and $N$ is the number of negative samples.

### 2.5 Recommender: Reason User Preferences and Fine-grained Ranking

The role of the recommender is to analyze user preferences based on the conversation history and the user feedback, and to rerank the candidate set. Due to the excellent language comprehension and reasoning capabilities of the Reasoning LLM, we use it as the foundation of the recommender and optimize it through a two-stage training paradigm.

#### 2.5.1 Two Stage Model Training

**SFT stage.** We first train the recommender to understand the task format and acquire basic preference reasoning ability. Given the instruction input $Q = \{Inst, D, I_{cand}, F\}$, where $Inst$ is the task description, $D$ is the conversation history, $I_{cand}$ is the candidate set retrieved by the retriever, and $F$ is the user feedback, the recommender is required to infer user preferences and rerank the candidate items. The output consists of a reasoning trajectory and a Top-$k$ recommendation list. To provide supervised signals, we use GPT-3.5-Turbo to generate the gold reasoning trajectory $T^*$ and recommendation list $R^*$, with additional access to the human-constructed list $R_D^{human} \in \{H^+, H^-, S^+, S^-\}$, the target item $i^*$, and its attributes $a^*$. We concatenate $T^*$ and $R^*$ as the target sequence $Y = [T^*; R^*]$. The recommender is optimized by minimizing the negative log-likelihood of generating $Y$ conditioned on $Q$:

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{(Q,Y)\sim\mathcal{D}_{sft}} \left[ -\sum_{j=1}^{|Y|} \log P_\theta\left(Y_j \mid Y_{<j}, Q\right) \right] \tag{13}$$

where $\mathcal{D}_{sft}$ denotes the SFT dataset, and $Y_j$ is the $j$-th token of the target output sequence.

**Multi-turn RL stage.** After the SFT stage, we further optimize the recommender with multi-turn RL. At turn $t$, the state $s_t = \left(D, I_D, \mathcal{C}_{t-1}, I_{\text{cand}}^t\right)$ comprises the conversation context, the mentioned entity set, the accumulated interaction history, and the retrieved candidate set. The action $o_t = (P_t, R_t)$ consists of the preference inference and the ranked recommendation list. After executing $o_t$, the simulator returns feedback $F_{t+1}$ and the retriever updates the candidate set, inducing the transition $s_{t+1} = \left(D, I_D, \mathcal{C}_t, I_{\text{cand}}^{t+1}\right)$. We optimize the

policy via a multi-turn extension of GRPO (Shao et al., 2024). In our method, simulator feedback $F_{t+1}$ is appended as a new user message at each turn, so a complete $T$-turn trajectory forms a single alternating dialogue sequence; only the recommender tokens $\{o_{i,t}\}$ contribute to the loss. For each query, $G$ trajectories are sampled from $\pi_{\theta_{\text{old}}}$. Let $N_i = \sum_{t=1}^{T_i} |o_{i,t}|$ be the total generated tokens in trajectory $i$, the group-relative advantage is $\hat{A}_i = (r_i - \mu_r)/\sigma_r$, where $r_i$ is the reward of the $i$-th trajectory, and $\mu_r$ and $\sigma_r$ denote the mean and standard deviation of the rewards within the same group, respectively. The optimization objective is formulated as follows:

$$\mathcal{J}(\theta) = \mathbb{E}\Bigg[\frac{1}{G}\sum_{i=1}^{G}\frac{1}{N_i}\sum_{t=1}^{T_i}\sum_{k=1}^{|o_{i,t}|}\Big(\min\Big(\rho_{i,t,k}\hat{A}_i,\ \text{clip}(\rho_{i,t,k}, 1-\epsilon, 1+\epsilon)\,\hat{A}_i\Big) - \beta\, D_{\text{KL}}\Big)\Bigg] \tag{14}$$

where $\rho_{i,t,k}$ is the token-level importance ratio. The $1/N_i$ normalization ensures equal contribution regardless of trajectory length. $D_{\text{KL}}$ denotes the Kullback–Leibler divergence between the updated policy $\pi_\theta$ and the reference policy $\pi_{\text{ref}}$.

#### 2.5.2 Reward Design

**Format reward.** To encourage structured reasoning, we apply two format checks. The process reward scores adherence to the five-step reasoning specification (preference inference, attribute matching, scoring, ranking, explanation) by counting regex matches $N_{\text{match}}$ within the `<think>` block:

$$r_{\text{think}} = \begin{cases} 1 - \frac{|N_{\text{match}} - 5|}{5}, & 1 \le N_{\text{match}} \le 7 \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

The answer reward checks whether the `<answer>` block matches the required format `RANK LIST: [Title, . . . , Title]`:

$$r_{\text{answer}} = \mathbf{1}[\text{match}(Ans)] \tag{16}$$

**Recommendation reward.** Given the target item $i^*$, we evaluate the ranked list $R_t$ via two complementary metrics. The hit reward $r_{\text{hit}} = \mathbf{1}[i^* \in R_t]$ checks presence, while the rank reward assigns higher rewards to higher-ranked target items through linear decay:

$$r_{\text{rank}} = \begin{cases} 1 - \dfrac{pos_{i^*} - 1}{|R_t|}, & i^* \in R_t \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

**Preference reward.** We measure whether the inferred preference $P_t$ aligns with the ground-truth preference $\hat{P}$ via semantic similarity:

$$r_{\text{pref}} = f_{\text{sim}}\Big(E_T(P_t),\ E_T(\hat{P})\Big) \tag{18}$$

where $f_{\text{sim}}$ is cosine similarity and $E_T$ is a text encoder.

## 3 Experiments

In this section, we attempt to answer the following research questions (RQs): RQ1: How does our method perform compared with existing CRS baselines? RQ2: Does training the user simulator and recommender improve recommendation performance? RQ3: Does the multi-turn preference optimization process lead to improvements in recommendation performance? RQ4: Where do the gains in recommendation performance come from, and does the system truly elicit user preferences?

### 3.1 Experimental Settings

In this section, we briefly introduce the experimental settings. A more detailed elaboration is presented in Appendix C.

**Dataset.** To verify the effectiveness of our method, we conduct experiments on two widely used CRS datasets, namely ReDial (Li et al., 2018) and INSPIRED (Hayati et al., 2020). ReDial contains 10,006 dialogues, 956 users, and 6,924 items. INSPIRED contains 1,001 dialogues, 1,482 users, and 1,123 items. We follow the dataset partitioning strategy used in previous works.

**Baselines.** (1) Conventional recommendation models: We include Item Popularity, which ranks items by their frequency, and BERT (Devlin et al., 2019). (2) General LLMs as zero-shot CRSs: Llama-3.1-8B-Instruct (Dubey et al., 2024) (L3.1-8B-I), GPT-3.5-Turbo and GPT-4o (Achiam et al., 2023). (3) Specialized CRS methods: ReDial (Li et al., 2018), KBRD (Chen et al., 2019), KGSF (Zhou et al., 2020), TREA (Li et al., 2023), VRICR (Zhang et al., 2023), UniCRS (Wang et al., 2022a), DCRS (Dao et al., 2024), and MSCRS (Wei et al., 2025).

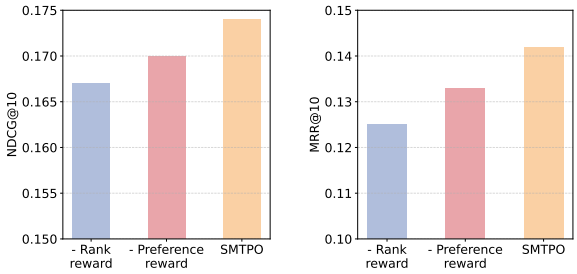
**Metrics.** Given the remarkable performance of LLMs in conversation tasks (Chang et al., 2024), we follow existing work (Xie et al., 2024; Wang et al., 2025) to focus our evaluation primarily on the recommendation task. We use Recall@k,

Table 1: Comparison of the main results between our method and baseline methods. The best result is given in bold. Significant improvements are marked with * (t-test, $p < 0.05$).

| Dataset | Metric | Popularity | BERT | L3.1-8B-I | GPT-3.5 | GPT-4o | ReDial | KBRD | KGSF | TREA | VRICR | UniCRS | DCRS | MSCRS | **SMTPO** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ReDial | Recall@1 | 0.011 | 0.027 | 0.037 | 0.043 | 0.047 | 0.010 | 0.033 | 0.035 | 0.045 | 0.054 | 0.065 | 0.076 | 0.081 | **0.092*** |
| | Recall@10 | 0.053 | 0.142 | 0.136 | 0.165 | 0.189 | 0.065 | 0.150 | 0.175 | 0.204 | 0.244 | 0.241 | 0.253 | 0.264 | **0.270*** |
| | NDCG@10 | 0.029 | 0.075 | 0.067 | 0.091 | 0.106 | 0.034 | 0.083 | 0.094 | 0.114 | 0.138 | 0.143 | 0.154 | 0.161 | **0.165*** |
| | MRR@10 | 0.021 | 0.055 | 0.046 | 0.069 | 0.077 | 0.024 | 0.062 | 0.070 | 0.087 | 0.106 | 0.113 | 0.123 | 0.128 | **0.130** |
| INSPIRED | Recall@1 | 0.031 | 0.049 | 0.043 | 0.053 | 0.079 | 0.009 | 0.042 | 0.051 | 0.047 | 0.043 | 0.085 | 0.093 | 0.096 | **0.099** |
| | Recall@10 | 0.155 | 0.189 | 0.162 | 0.177 | 0.196 | 0.048 | 0.135 | 0.132 | 0.146 | 0.141 | 0.230 | 0.226 | 0.257 | **0.273*** |
| | NDCG@10 | 0.085 | 0.112 | 0.097 | 0.109 | 0.121 | 0.023 | 0.088 | 0.092 | 0.095 | 0.091 | 0.149 | 0.153 | 0.168 | **0.174*** |
| | MRR@10 | 0.064 | 0.088 | 0.077 | 0.088 | 0.098 | 0.015 | 0.073 | 0.079 | 0.076 | 0.075 | 0.125 | 0.130 | 0.140 | **0.142*** |

Table 2: Ablation study of the simulator and the recommender on INSPIRED dataset. The best result is given in bold. Significant improvements are marked with * (t-test, $p < 0.05$).

| Method | Recall@1 | Recall@10 | NDCG@10 | MRR@10 |
|---|---|---|---|---|
| w/o RL | 0.075 | 0.203 | 0.129 | 0.107 |
| w/o SFT | 0.056 | 0.187 | 0.108 | 0.084 |
| w/o Pref | 0.082 | 0.217 | 0.144 | 0.122 |
| w/o Rec | 0.076 | 0.207 | 0.136 | 0.113 |
| w/ Single-task | 0.095 | 0.220 | 0.152 | 0.130 |
| w/o Feedback | 0.072 | 0.197 | 0.126 | 0.105 |
| SMTPO | **0.099*** | **0.273*** | **0.174*** | **0.142*** |




(a) NDCG (b) MRR

Figure 3: The performance impact of different rewards on the INSPIRED dataset.

NDCG@k, and MRR@k for evaluation, with $k \in \{1, 10\}$. We clarify that during inference, each dialogue instance in the test set is independently run for the full multi-turn interaction (up to five turns). For each turn $t$, recommendation metrics are computed independently on the same fixed and complete test set, rather than on the subset of dialogues that have not yet reached the target.

### 3.2 Overall Performance Analysis

To answer RQ1, we compare SMTPO with representative baselines in Table 1. Overall, traditional recommenders and zero-shot general LLMs show limited performance. Zero-shot LLMs underperform specialized CRS models, highlighting the necessity of task-specific optimization. Specialized CRS methods achieve stronger results by exploiting richer contextual signals. For example, UniCRS and DCRS benefit from entity-aware representations and prompt-based optimization, while DCRS further benefits from contextual retrieval. MSCRS shows the value of multi-modal features and semantic graphs for preference modeling.

Our proposed SMTPO consistently outperforms all baselines, mainly due to its feedback-aware and preference-oriented optimization. The instruction-tuned simulator provides informative feedback without relying on ground-truth labels during inference, reducing misleading signals during inference. Meanwhile, the recommender is trained through a two-stage SFT-RL process, where fine-grained rewards guide multi-turn preference alignment and improve robustness against imperfect feedback.

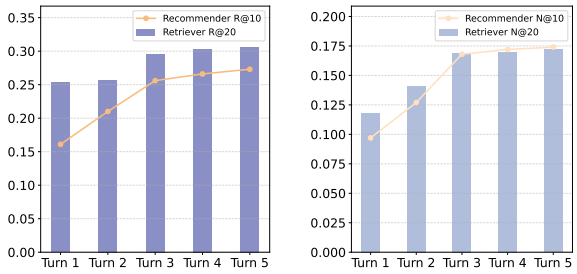
### 3.3 Ablation: Recommender and Simulator

To answer RQ2, we conduct ablation studies on the INSPIRED dataset to evaluate the effects of recommender training and simulator training.

We first evaluate the recommender module with four variants: **w/o RL**, trained only with SFT; **w/o SFT**, trained only with RL; **w/o Pref**, excluding inferred preferences from retrieval; and **w/o Rec**, excluding the recommendation list from retrieval. As shown in Table 2, removing SFT leads to the largest drop, with Recall@1 decreasing from 0.099 to 0.056, showing that SFT is crucial for warm-starting the recommender. Removing RL also degrades performance, showing that the gains are not from simple prompting or reranking, but from reward-based policy optimization under simulator feedback. The declines of w/o Pref and w/o Rec further show that preference inference and recommendation results provide useful multi-turn signals for dynamic retrieval.

Table 3: Blind-simulator and Noisy-feedback test on SMTPO. The best result is given in bold. Significant improvements are marked with * (t-test, $p < 0.05$).

| R@10 | Turn 1 | Turn 2 | Turn 3 | Turn 4 | Turn 5 |
|---|---|---|---|---|---|
| Blind Simulator | 0.161 | 0.194 | 0.171 | 0.164 | 0.167 |
| Noise Test | 0.161 | 0.182 | 0.185 | 0.178 | 0.181 |
| SMTPO | **0.161** | **0.210*** | **0.256*** | **0.266*** | **0.273*** |




(a) Recall (b) NDCG

Figure 4: Impact of multi-turn preference optimization on recommendation performance.

We then analyze the reward design in multi-turn RL. Since the format reward and hit reward provide valid interaction constraints and basic optimization signals, we keep them in all variants and ablate rank and preference rewards. Figure 3 shows that removing any additional reward degrades performance, among which removing the rank reward causes the largest drop, highlighting its role in improving reranking ability.

Finally, we evaluate the simulator module with two variants: **w/ Single-task**, fine-tuned only for feedback generation, and **w/o Feedback**, which removes natural language feedback during candidate generation. As shown in Table 2, single-task SFT underperforms multi-task SFT, suggesting that multi-task training helps generate more informative feedback. The further decline of w/o Feedback confirms the importance of natural language feedback for candidate selection.

### 3.4 Multi-Turn Optimization

To answer RQ3, we analyze SMTPO's performance across conversation turns, as shown in Figures 4a and 4b. The recommender's R@10 and N@10 steadily improve with more turns, indicating that multi-turn interactions help refine user preferences. Notably, its N@10 eventually surpasses the retriever's N@20, showing the effectiveness of reranking. Overall, the turn-wise trajectory verifies that SMTPO can gradually align preference inference and ranking decisions with user interests, improving both accuracy and robustness.

### 3.5 Source of Performance Gains

To answer RQ4, we examine whether dynamic multi-turn inputs are necessary. First, we conduct a blind simulator experiment, where the simulator only observes conversation history without the previous recommendation list $R_{t-1}$. As shown in Table 3, performance drops significantly, indicating that SMTPO benefits from dynamic interaction modeling rather than pretraining memorization. We further replace simulator feedback with irrelevant conversational utterances. The resulting degradation shows that SMTPO truly learns from meaningful feedback signals, rather than relying on prompt engineering or memorization.

Overall, the multi-turn preference optimization mechanism gradually builds a reliable estimate of users' implicit preferences within a few turns. During training, the recommender uses multi-view rewards to align conversation and simulator feedback with target items. At inference, it updates user preferences from feedback-driven implicit preference posteriors, enabling robust preference elicitation and improved recommendation performance.

## 4 Conclusion

This work presents SMTPO, a simulator-guided multi-turn preference optimization framework for conversational recommendation. SMTPO addresses the limitations of prior methods that mainly rely on simulator feedback for evaluation or candidate filtering by converting potentially biased feedback into policy-level optimization signals. Specifically, a multi-task user simulator provides informative feedback, while a reasoning LLM-based recommender is first trained with SFT and then optimized via multi-turn RL with fine-grained rewards to progressively align preference inference and ranking decisions with true user preferences. A dual semantic-collaborative retriever further constrains candidate generation within the global item space. Experiments show that SMTPO improves recommendation performance, robustness, and generalization, demonstrating the effectiveness of preference-oriented policy optimization under imperfect simulator feedback.

## Limitations

SMTPO is a novel multi-turn preference optimization framework that uses a user simulator to guide a

Reasoning LLM recommender in aligning with true user preferences. Its performance heavily depends on the quality of simulated feedback. We train the simulator via multi-task SFT, including feedback generation, attribute alignment, and target prediction, to produce realistic and diverse responses. Experiments show that multi-task training improves feedback quality compared to single-task versions, but gaps between simulated and real users remain, which may cause distribution shifts in deployment.

Nevertheless, (1) noise-injection experiments indicate that the recommender's gains arise from continuous alignment with true preference signals rather than prompt engineering or memorization, suggesting some robustness to feedback quality; (2) reducing the gap between simulated and real users remains a key challenge in multi-turn conversational recommendation. Overall, SMTPO provides a meaningful contribution, and developing high-quality simulators that better reflect real user behavior is a promising direction for future work.

## References


Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, and 1 others. 2024. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology*, 15(3):1–45.

Qibin Chen, Junyang Lin, Yichang Zhang, Ming Ding, Yukuo Cen, Hongxia Yang, and Jie Tang. 2019. Towards knowledge-based recommender dialog system. *arXiv preprint arXiv:1908.05391*.

Huy Dao, Yang Deng, Dung D Le, and Lizi Liao. 2024. Broadening the view: Demonstration-augmented prompt learning for conversational recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 785–795.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

Yang Deng, Yaliang Li, Fei Sun, Bolin Ding, and Wai Lam. 2021. Unified conversational recommendation policy learning via graph-based reinforcement learning. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1431–1441.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

Xiaofei Dong, Xueqiang Zhang, Weixin Bu, Dan Zhang, and Feng Cao. 2024. A survey of llm-based agents: Theories, technologies, applications and suggestions. In *2024 3rd International Conference on Artificial Intelligence, Internet of Things and Cloud Computing Technology (AIoTC)*, pages 407–413. IEEE.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Yi Fang, Wenjie Wang, Yang Zhang, Fengbin Zhu, Qifan Wang, Fuli Feng, and Xiangnan He. 2025. Reason4rec: Large language models for recommendation with deliberative user preference alignment. *arXiv preprint arXiv:2502.02061*.

Xueyang Feng, Jingsen Zhang, Jiakai Tang, Wei Li, Guohao Cai, Xu Chen, Quanyu Dai, Yue Zhu, and Zhenhua Dong. 2025. Expectation confirmation preference optimization for multi-turn conversational recommendation agent. *arXiv preprint arXiv:2506.14302*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Shirley Anugrah Hayati, Dongyeop Kang, Qingxiaoyang Zhu, Weiyan Shi, and Zhou Yu. 2020. Inspired: Toward sociable recommendation dialog systems. *arXiv preprint arXiv:2009.14306*.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 639–648.

Zhankui He, Zhouhang Xie, Rahul Jha, Harald Steck, Dawen Liang, Yesu Feng, Bodhisattwa Prasad Majumder, Nathan Kallus, and Julian McAuley. 2023. Large language models as zero-shot conversational recommenders. In *Proceedings of the 32nd ACM international conference on information and knowledge management*, pages 720–730.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. 2024. Understanding the planning of llm agents: A survey. *arXiv preprint arXiv:2402.02716*.

Wenqiang Lei, Xiangnan He, Yisong Miao, Qingyun Wu, Richang Hong, Min-Yen Kan, and Tat-Seng Chua. 2020a. Estimation-action-reflection: Towards deep interaction between conversational and recommender systems. In *Proceedings of the 13th international conference on web search and data mining*, pages 304–312.

Wenqiang Lei, Gangyi Zhang, Xiangnan He, Yisong Miao, Xiang Wang, Liang Chen, and Tat-Seng Chua. 2020b. Interactive path reasoning on graph for conversational recommendation. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2073–2083.

Raymond Li, Samira Ebrahimi Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. 2018. Towards deep conversational recommendations. *Advances in neural information processing systems*, 31.

Wendi Li, Wei Wei, Xiaoye Qu, Xian-Ling Mao, Ye Yuan, Wenfeng Xie, and Dangyang Chen. 2023. Trea: Tree-structure reasoning schema for conversational recommendation. *arXiv preprint arXiv:2307.10543*.

Ying-Chun Lin, Jennifer Neville, Jack W Stokes, Longqi Yang, Tara Safavi, Mengting Wan, Scott Counts, Siddharth Suri, Reid Andersen, Xiaofeng Xu, and 1 others. 2024. Interpretable user satisfaction estimation for conversational systems with large language models. *arXiv preprint arXiv:2403.12388*.

Zhangchi Qiu, Ye Tao, Shirui Pan, and Alan Wee-Chung Liew. 2024. Knowledge graphs and pretrained language models enhanced representation learning for conversational recommender systems. *IEEE transactions on neural networks and learning systems*.

Wasswa Shafik. 2024. Introduction to chatgpt. In *Advanced applications of generative AI and natural language processing models*, pages 1–25. IGI Global Scientific Publishing.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

Xiaolei Wang, Xinyu Tang, Wayne Xin Zhao, Jingyuan Wang, and Ji-Rong Wen. 2023. Rethinking the evaluation for conversational recommendation in the era of large language models. *arXiv preprint arXiv:2305.13112*.

Xiaolei Wang, Chunxuan Xia, Junyi Li, Fanzhe Meng, Lei Huang, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. 2025. Search-based interaction for conversation recommendation via generative reward model based simulated user. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 75–84.

Xiaolei Wang, Kun Zhou, Ji-Rong Wen, and Wayne Xin Zhao. 2022a. Towards unified conversational recommender systems via knowledge-enhanced prompt learning. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pages 1929–1937.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022b. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Yuling Wang, Changxin Tian, Binbin Hu, Yanhua Yu, Ziqi Liu, Zhiqiang Zhang, Jun Zhou, Liang Pang, and Xiao Wang. 2024. Can small language models be good reasoners for sequential recommendation? In *Proceedings of the ACM Web Conference 2024*, pages 3876–3887.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Yibiao Wei, Jie Zou, Weikang Guo, Guoqing Wang, Xing Xu, and Yang Yang. 2025. Mscrs: Multi-modal semantic graph prompt learning framework for conversational recommender systems. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 42–52.

Junda Wu, Cheng-Chun Chang, Tong Yu, Zhankui He, Jianing Wang, Yupeng Hou, and Julian McAuley. 2024. Coral: collaborative retrieval-augmented large language models improve long-tail recommendation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3391–3401.

Yunjia Xi, Weiwen Liu, Jianghao Lin, Bo Chen, Ruiming Tang, Weinan Zhang, and Yong Yu. 2024. Memocrs: Memory-enhanced sequential conversational recommender systems with large language models.

In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 2585–2595.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pages 641–649.

Zhouhang Xie, Junda Wu, Hyunsik Jeon, Zhankui He, Harald Steck, Rahul Jha, Dawen Liang, Nathan Kallus, and Julian McAuley. 2024. Neighborhood-based collaborative filtering for conversational recommendation. In *Proceedings of the 18th ACM Conference on Recommender Systems*, pages 1045–1050.

Kerui Xu, Jingxuan Yang, Jun Xu, Sheng Gao, Jun Guo, and Ji-Rong Wen. 2021. Adapting user preference to online feedback in multi-round conversational recommendation. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 364–372.

Ting Yang and Li Chen. 2024. Unleashing the retrieval potential of large language models in conversational recommender systems. In *Proceedings of the 18th ACM Conference on Recommender Systems*, pages 43–52.

Se-eun Yoon, Zhankui He, Jessica Maria Echterhoff, and Julian McAuley. 2024. Evaluating large language models as generative user simulators for conversational recommendation. *arXiv preprint arXiv:2403.09738*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, and 1 others. 2025. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*.

Pengfei Zhang, Guojia An, Jin Huang, Yuhan Yang, Yang Yang, and Jie Zou. 2026. Not all information brings benefits: Personalization-driven agent debate for conversational recommendation. In *Proceedings of the ACM Web Conference 2026*, pages 5843–5852.

Xiaoyu Zhang, Xin Xin, Dongdong Li, Wenxuan Liu, Pengjie Ren, Zhumin Chen, Jun Ma, and Zhaochun Ren. 2023. Variational reasoning over incomplete knowledge graphs for conversational recommendation. In *Proceedings of the sixteenth ACM international conference on web search and data mining*, pages 231–239.

Keyu Zhao, Fengli Xu, and Yong Li. 2025. Reason-to-recommend: Using interaction-of-thought reasoning to enhance llm recommendation. *arXiv preprint arXiv:2506.05069*.

Bowen Zheng, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, Ming Chen, and Ji-Rong Wen. 2024a. Adapting large language models by integrating collaborative semantics for recommendation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 1435–1448. IEEE.

Yongsen Zheng, Mingjie Qian, Guohua Wang, Yang Liu, Ziliang Chen, Mingzhi Mao, Liang Lin, and Kwok-Yan Lam. 2025a. Hypercrs: Hypergraph-aware multi-grained preference learning to burst filter bubbles in conversational recommendation system. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 5597–5608.

Yongsen Zheng, Zongxuan Xie, Guohua Wang, Ziyao Liu, Liang Lin, and Kwok-Yan Lam. 2025b. Why multi-interest fairness matters: Hypergraph contrastive multi-interest learning for fair conversational recommender system. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 25673–25684.

Yongsen Zheng, Ruilin Xu, Ziliang Chen, Guohua Wang, Mingjie Qian, Jinghui Qin, and Liang Lin. 2024b. Hycorec: Hypergraph-enhanced multi-preference learning for alleviating matthew effect in conversational recommendation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2526–2537.

Kun Zhou, Wayne Xin Zhao, Shuqing Bian, Yuanhang Zhou, Ji-Rong Wen, and Jingsong Yu. 2020. Improving conversational recommender systems via knowledge graph based semantic fusion. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1006–1014.

Yaochen Zhu, Chao Wan, Harald Steck, Dawen Liang, Yesu Feng, Nathan Kallus, and Jundong Li. 2025. Collaborative retrieval for large language model-based conversational recommender systems. In *Proceedings of the ACM on Web Conference 2025*, pages 3323–3334.

Yaochen Zhu, Liang Wu, Qi Guo, Liangjie Hong, and Jundong Li. 2024. Collaborative large language model for recommender systems. In *Proceedings of the ACM Web Conference 2024*, pages 3162–3172.

# A Related Work

**LLM-based Conversational Recommender Systems (CRSs).** Existing CRS research can be broadly categorized into attribute-based CRSs (Deng et al., 2021; Xu et al., 2021; Lei et al., 2020a,b) and generation-based CRSs (Zheng et al., 2025b,a; Dao et al., 2024; Li et al., 2023; Wang et al., 2022a). Attribute-based CRSs rely on fixed templates, leading to mechanical interaction patterns, while generation-based CRSs combine pre-trained language models (PLMs) with knowledge graphs but still struggle to model diverse user preferences. Given the powerful semantic understanding capabilities of LLMs (Wang et al., 2024; Wei et al., 2022), recent studies have introduced them into CRS to address these limitations. For example, He et al. (2023) systematically analyzed the performance of LLMs in zero-shot CRS; Xi et al. (2024) proposed MemoCRS, which leverages a memory-augmented LLM to manage users' historical preferences, thereby enhancing the personalization of recommendations; Yang and Chen (2024) unified the LLM into a CRS with both retrieval and generation capabilities through instruction tuning. LLMs not only possess strong language understanding and generation capabilities but can also exhibit multi-step reasoning abilities through reinforcement learning (RL) (Yu et al., 2025; Yue et al., 2025) or multi-path sampling (Brown et al., 2024; Wang et al., 2022b). RL-based post-training methods, such as DeepSeek-R1 (Guo et al., 2025) and Kimi k1.5 (Team et al., 2025), enable user preference analysis and candidate item evaluation. Although the reasoning capabilities of LLMs have been validated in structured tasks, their potential in CRS remains largely unexplored.

**User Simulator for CRSs.** Although LLMs bring new opportunities to CRSs, they often struggle to model user preferences with limited context, and frequent reliance on real users increases interaction costs. To address this, prior studies introduced user simulators that generate natural language feedback to assist recommender optimization (Feng et al., 2025; Yoon et al., 2024; Wang et al., 2023; Lin et al., 2024). Early works mainly used simulators for evaluation with ground-truth preference labels (Yoon et al., 2024; Wang et al., 2023), making them difficult to apply directly in real interaction scenarios. Wang et al. (2025) were the first to propose a simulator that does not require preference labels, enabling automatic interaction with CRSs. However, due to the lack of a proper supervision mechanism, these methods become unreliable when the simulator's feedback is biased. The generalization ability of the recommender still needs improvement.

Our method belongs to the LLM-based CRS and introduces a user simulator. Unlike previous methods, SMTPO employs a label-free simulator to generate high-quality natural language feedback and adopts a Reasoning LLM as the recommender backbone, leveraging its strong multi-step reasoning ability and RL algorithm, SMTPO achieves multi-turn preference optimization to filter noisy feedback, and significantly enhances recommendation performance.

**Algorithm 1:** Hard Negative Sampling

**Input:** Target items $T$, all items $I$, attribute map $A$, number of hard negatives $k$
**Output:** Hard negative set $H$

```
I ← I \ T // Remove target items
H ← ∅ // Initialize hard negative set
T_attr ← ∅ // All attributes of target items
for t ∈ T do
    T_attr ← T_attr ∪ A(t)
for i ∈ I do
    score(i) ← |A(i) ∩ T_attr|
    // Count score
I ← Sort(I, by score(i), descending)
// Sort by score
H ← top-k items from I // Select hard negatives
return H
```

# B Prompts

## B.1 Data Construction Prompt

To support the training of the recommender and simulator, we use GPT-3.5-Turbo to construct two types of instruction data.

The first type is designed for recommender fine-tuning. Specifically, GPT-3.5-Turbo is prompted as a professional movie recommendation expert and provided with the conversation history $D$, candidate set $R_D^{\text{human}}$, user target item $i^*$, and its attributes $a^*$. The model is required to follow a five-step reasoning process, including preference

inference, attribute matching, scoring, ranking and selection, and explanation generation. To avoid information leakage, the reasoning process is constrained to rely only on the conversation history, without directly using the target item. The model then outputs the top-10 recommendations in JSON format, which are expected to be consistent with the user's true preferences. The corresponding input and output are shown in Figure 5.

The second type is used for user simulator fine-tuning. In this setting, GPT-3.5-Turbo is prompted as a user familiar with movie attributes and generates informative natural language feedback based on the conversation history $D$, candidate set $R_D^{\text{human}}$, target item $i^*$, and attributes $a^*$. The corresponding input and output are shown in Figure 6.

### B.2 Supervised Fine-Tuning (SFT) Prompt

We conduct SFT separately for the user simulator and the recommender. For the user simulator, we design three tasks:

Feedback Generation Task: The simulator generates natural and informative feedback as a user, based on the conversation history and given recommendation list. The GPT-3.5-Turbo generated simulated feedback is used as fine-tuning data. Instruction is shown in Figure 7. The entity information $I_D$ was included in {Conversation History} in the prompt.

Attribute Alignment Task: The simulator identifies the user's target movies and key attributes from conversation history. This helps the simulator learn user preferences. Fine-tuning data is constructed from the original dataset, with movie attributes extracted from the KG $\mathcal{G}$. The instruction is shown in Figure 8.

Target Prediction Task: The simulator performs binary classification (strictly "Yes" or "No") on a recommended movie. Positive samples are movies matching the user's true preferences, and hard negatives are generated using Algorithm 1. The instruction is shown in Figure 9.

For recommender fine-tuning, it is required to reason within **<think>...</think>** tags following the five-step process and output Top-10 recommendations in the format **RANK LIST: [...]**. Instruction is shown in Figure 10. This design ensures the recommender learns interpretable preference modeling and recommendation capabilities, forming the basis for subsequent multi-turn RL stage and reward design.

## C Experimental Details

For user simulator, we adopt Llama-3.1-8B-Instruct as the backbone. The model is SFT-trained for 5 epochs with a learning rate of $1 \times 10^{-5}$, LoRA rank 8, and the AdamW optimizer. For retriever, training runs up to 50 epochs with early stopping, a learning rate of $5 \times 10^{-5}$, and the Adam optimizer. The top-20 candidates are retrieved using bge-base-en-v1.5 (Xiao et al., 2024) for text encoding and LightGCN (He et al., 2020) for collaborative encoding. For recommender, we adopt DeepSeek-R1-Distill-Llama-8B (Guo et al., 2025). It is SFT-trained for 8 epochs (learning rate $5 \times 10^{-5}$, LoRA rank 8, AdamW), followed by multi-turn GRPO optimization for 3 epochs. The group size of GRPO is 4, with gradient accumulation 8, max length 2048, learning rate $1 \times 10^{-5}$, and 5 interaction turns. FlashAttention2 (Dao, 2023) accelerates training. The recommender receives 20 candidates and outputs a final top-10 list.

## D Case Study

We illustrate a multi-turn episode from INSPIRED in Table 4. The dialogue history is dominated by superhero preferences, while the ground-truth target is Joker (2019), a psychological crime drama. This mismatch is representative of simulator-guided CRS: feedback can remain context-plausible yet still misaligned with the latent target.

At Turn 0, the recommender correctly interprets the surface dialogue and ranks superhero titles at the top; Joker appears in the retrieved pool but not in the final Top-10. In subsequent turns, the simulator continues to provide detailed superhero-oriented feedback conditioned on the history and the previous list. Rather than passively following this signal, the recommender progressively revises its preference inference toward darker, character-driven crime themes and promotes Joker from outside the Top-10 to #3 and then #1. Throughout, the dual-view retriever keeps recalling Joker, so improvement comes from policy-level reranking under imperfect feedback, not from filtering candidates by the simulator. This case therefore complements our main claim: SMTPO learns to extract reliable optimization signals across turns even when simulator utterances do not fully reveal the true target preference.

Table 4: Case study about multi-turn preference optimization on the INSPIRED dataset.

| **Base Info** | |
|---|---|
| **Target Item** | Joker(2019) "category": [drama film,psychological thriller film,crime film]; "director": Todd Phillips |
| **History** | **user:** I'm looking for something superhero-related, like *Guardians of the Galaxy*. **recommender:** Are you interested in comic-book characters? **user:** Yes, I love characters like Spider-Man, Iron Man, and Captain America. **recommender:** Would you like to watch the trailer for *The Avengers*? **user:** Yes, that sounds great. **recommender:** That's the movie with Thanos, right? **user:** Yes, Thanos is in it. **recommender:** Superhero team-ups are always fun. **user:** Agreed, these superhero mashups have been great. |
| **Turn 0(Cold Start)** | |
| **Candidates(I)** | 'Avengers (2012)', 'Avengers: Endgame (2019)', 'Marvel (2008)', 'Batman (1989)', 'Black Panther (2018)', 'Deadpool (2016)', 'Black Widow (2020)', 'Joker (2019)', 'Avenger (2006)', 'Dark Phoenix (2019)', 'Avengers: Infinity War (2018)', 'Captain Marvel (2019)', 'Spider-Man: Homecoming 2 (2019)', 'The Avengers (2012)', 'Thor: Ragnarok (2017)', 'Aquaman (2018)', 'Thor (2011)', 'Justice League (2015)', 'Hulk (2003)', 'Suicide Squad (2016)' |
| **Reasoning(O)** | <think>**Step1:Preference Inference.** The user has expressed a strong interest in action movies, particularly those involving superhero characters. They mentioned loving action movies and specifically asked for something superhero-related, like 'Guardians of the Galaxy'. They also mentioned enjoying comic book characters such as Spider-Man, Iron Man, and Captain America. The user showed enthusiasm for the Avengers, particularly the presence of Thanos, indicating that he prefers thrilling superhero movies. **Step2:**...</think> |
| **RecList(O)** | 1. Avengers (2012), 2. Avengers: Endgame (2019), 3. Avengers: Infinity War (2018), 4. Black Panther (2018), 5. Thor: Ragnarok (2017), 6. Captain Marvel (2019), 7. Deadpool (2016), 8. Black Widow (2020), 9. Spider-Man: Homecoming 2 (2019), 10. Aquaman (2018) |
| **Turn 1(Warm Up)** | |
| **Candidates(I)** | 'Star Wars (1977)', 'Avengers (2012)', 'Black Panther (2018)', 'Marvel (2008)', 'Avengers: Infinity War (2018)', 'Avengers: Endgame (2019)', 'Star Wars: The Rise of Skywalker (2019)', 'The Avengers (2012)', 'Joker (2019)', 'Captain America (2011)', 'Blade Runner (1982)', 'Captain Marvel (2019)', 'Batman (1989)', 'Avenger (2006)', 'Blade Runner 2049 (2017)', 'Spider-Man: Homecoming 2 (2019)', 'Star Wars: The Last Jedi (2017)', 'Deadpool (2016)', 'Thor: Ragnarok (2017)', 'Black Widow (2020)' |
| **Feedback(I)** | I really enjoy the team-up dynamics in superhero films, especially when they involve epic battles against a major villain like Thanos. The Russo brothers' direction and Marvel Studios' productions always deliver that perfect blend of action and character depth. I'm particularly interested in movies where the heroes have to overcome their differences and work together against an overwhelming threat, with a mix of sci-fi elements and emotional stakes. Something with a strong ensemble cast and a balance of humor and intensity would be ideal. |
| **Reasoning(O)** | <think>**Step1:Preference Inference.**The user is looking for a movie that is intense, character-driven, and features deep themes, particularly those involving psychological drama, societal issues, and a gripping narrative. The user prefers films with a dark, atmospheric setting and strong performances, especially those that explore the unraveling of a protagonist's psyche. Additionally, the user appreciates crime elements, and prefers big studio productions that can deliver this vibe. **Step2:**...</think> |
| **RecList(O)** | 1.Blade Runner (1982), 2.Blade Runner 2049 (2017), 3.Joker (2019), 4.Batman (1989), 5.Avenger (2006), 6.The Last Jedi (2017), 7.Star Wars (1977), 8.Avengers (2012), 9.Black Panther (2018), 10.Avengers: Infinity War (2018) |
| **Turn 2(Warm Up)** | |
| **Candidates(I)** | 'Star Wars (1977)', 'Avengers (2012)', 'Black Panther (2018)', 'Avengers: Infinity War (2018)', 'Marvel (2008)', 'Avengers: Endgame (2019)', 'Blade Runner (1982)', 'Star Wars: The Rise of Skywalker (2019)', 'Avenger (2006)', 'The Last Jedi (2017)', 'The Avengers (2012)', 'Captain America (2011)', 'Batman (1989)', 'Blade Runner 2049 (2017)', 'Joker (2019)', 'Dark Phoenix (2019)', 'Captain Marvel (2019)', 'Black Widow (2020)', 'Deadpool (2016)', 'John Wick (2014)' |
| **Feedback(I)** | I really enjoy superhero team-up movies with high stakes, especially ones where the characters have to overcome their differences to face a big threat like Thanos. I love the mix of action, sci-fi, and humor, and directors who can balance all those elements well. A film with a strong ensemble cast—maybe including Iron Man, Captain America, and the Hulk—would be perfect. Also, I prefer Marvel Studios productions since they tend to deliver that perfect blend of character dynamics and epic battles. Something from the Russo brothers or similar directors would be great, since they handle large-scale superhero stories well. |
| **Reasoning(O)** | <think>**Step1:Preference Inference.**The user are looking for crime dramas with thriller elements and character driven elements. . They specifically mentioned enjoying films that explore complex characters, particularly those dealing with mental turmoil or societal issues, and appreciate a dark, atmospheric setting. The user also values bold storytelling and strong performances, with a preference for big studio productions that often nail this vibe. The user's interest in crime elements and a slow-burn but powerful payoff is also notable.**Step2:**...</think> |
| **RecList(O)** | 1.Joker (2019), 2.Blade Runner 2049 (2017), 3.Blade Runner (1982), 4.Batman (1989), 5.Avengers (2012), 6.Avengers: Infinity War (2018), 7.Black Panther (2018), 8.The Avengers (2012), 9.Captain America (2011), 10.Star Wars (1977) |

You are a professional movie recommendation expert. You are skilled at inferring user preferences from conversation history and are familiar with movie attributes such as cast (actors), genre, director, and production companies.
Your task is to conduct systematic thinking based on the following information and generate a final ranked recommendation list.
【Conversation History】
{conversation_history}
【Candidate Pool】
{candidate_pool}
【User Preferences】
{liked_genres}
【Task Instructions】
1. Your output must include two parts: **Reason** and **Recommend**.
- Reason contains the reasoning process. Your reasoning process must be as detailed as possible. You must output all five reasoning steps as below.
- Recommend contains the final recommended movie list.
2. The Reason section must strictly follow the requirements of each step as below. Each step should be clearly written as instructed, with logical reasoning, and the content should not be brief:
- **Step 1**: Preference Inference — Based only on the 【Conversation History】, infer the user's potential preferences, including attributes such as cast, genre, director, production company, or any other implicit interests.
- **Step 2**: Attribute Matching — Analyze which movies and their attributes in the 【Candidate Pool】 match the inferred preferences from Step 1.
- **Step 3**: Scoring — Assign each movie in the 【Candidate Pool】 a score based on how well it matches the inferred user preferences. Scores should range from 0.00 to 5.00, accurate to two decimal places.
- **Step 4**: Ranking and Selection — Rank all movies by their scores and select the top 10 as the final recommendations. The top-ranked items must cover the genres listed in 【User Preferences】, even though these preferences should not be used directly for inference.
- **Step 5**: Explanation — Provide a concise explanation of the final recommendations. You do not need to explain each movie individually—just summarize how the recommendations align with the inferred user preferences and key movie attributes.
Important: Your Step 1, Step 2, Step 3, Step 4 and Step 5 must be detailed and relatively lengthy. It is strictly forbidden to use the 【User Preferences】 for inference in Step 1, Step2 and Step 5. You must infer preferences solely from the 【Conversation History】. However, the final recommendations should still align with the true preferences for evaluation purposes.
3. The Recommend section must list the 10 selected movie titles from Step 4 in the following format: "[movie1, movie2, ..., movie10]"
4. Your output must be in **JSON** format:
{{ "Step1": "...", "Step2": "...", "Step3": "...", "Step4": "...", "Step5": "...", "Recommend": "..." }}

***ChatGPT Input***

{
"**Step 1**": "Preference Inference — Based on the 【Conversation History】, the user's preferences can be inferred as follows: The user explicitly mentions a liking for action movies, with 'Terminator (1984)' as a favorite, but indicates that the sci-fi element is not a significant factor. They also express interest in drama movies and show enthusiasm for a revenge-themed plot. The user prefers more recent films and is comfortable with violence. Additionally, they have a positive opinion of Keanu Reeves. This suggests a preference for action-packed, violent, revenge-themed films with strong lead actors, particularly those starring Keanu Reeves, and a secondary interest in drama elements.",
"**Step 2**": "Attribute Matching — Analyzing the 【Candidate Pool】, the following movies match the inferred preferences: 'John Wick (2014)' and 'John Wick: Chapter 3 - Parabellum (2019)' both feature Keanu Reeves, are action-packed, violent, and recent. 'The Accountant (2016)' combines action, crime, and drama with a strong lead. 'Kill the Irishman (2011)' is a mix of action, crime, and drama with a revenge element. 'Source Code (2011)' and 'Annihilation (2018)' have action and sci-fi elements, though the user is indifferent to sci-fi. 'Showtime (2002)' and 'Tropic Thunder (2008)' are action-comedies, which might not fully align with the user's preference for serious action. 'Rollercoaster (1977)' is too old and doesn't fit the recent preference. The rest either lack action or are not recent enough.",
"**Step 3**": "Scoring — Assigning scores based on matching inferred preferences: 'John Wick (2014)' - 5.00 (perfect match with action, violence, Keanu Reeves, recent), 'John Wick: Chapter 3 - Parabellum (2019)' - 4.80 (similar to the first but slightly less recent), 'The Accountant (2016)' - 4.50 (action, drama, recent), 'Kill the Irishman (2011)' - 4.20 (action, drama, revenge), 'Source Code (2011)' - 3.50 (action, sci-fi, recent), 'Annihilation (2018)' - 3.20 (action, sci-fi, recent), 'Showtime (2002)' - 2.50 (action-comedy, not recent), 'Tropic Thunder (2008)' - 2.00 (action-comedy, not recent), 'Rollercoaster (1977)' - 1.00 (too old, not matching recent preference), others - 0.00 (not matching key preferences).",
"**Step 4**":"Ranking and Selection — Ranking by scores: 1. 'John Wick (2014)' (5.00), 2. 'John Wick: Chapter 3 - Parabellum (2019)' (4.80), 3. 'The Accountant (2016)' (4.50), 4. 'Kill the Irishman (2011)' (4.20), 5. 'Source Code (2011)' (3.50), 6. 'Annihilation (2018)' (3.20), 7. 'Showtime (2002)' (2.50), 8. 'Tropic Thunder (2008)' (2.00), 9. 'Rollercoaster (1977)' (1.00), 10. 'Queen & Slim (2019)' (0.50, slight drama match).",
"**Step 5**": "Explanation — The final recommendations prioritize action-packed, violent, and revenge-themed films with strong leads, particularly those starring Keanu Reeves, aligning with the user's expressed preferences. The inclusion of 'The Accountant' and 'Kill the Irishman' caters to the drama interest, while 'John Wick' series perfectly matches the action and actor preference. 'Source Code' and 'Annihilation' are included for their action elements despite the sci-fi aspect, as they are recent and well-rated. The list ensures a balance between the user's primary and secondary genre preferences."
"**Recommend**": ["John Wick (2014)", "John Wick: Chapter 3 - Parabellum (2019)", "The Accountant (2016)", "Kill the Irishman (2011)", "Source Code (2011)", "Annihilation (2018)", "Showtime (2002)", "Tropic Thunder (2008)", "Rollercoaster (1977)", "Queen & Slim (2019)"]
}

***ChatGPT Output***

Figure 5: prompt reason

You are a knowledgeable user of an intelligent movie recommendation system, familiar with key movie attributes such as cast, genre, director, and production companies. Your response should be natural and detailed, and informative, helping the system better understand your preferences and improve future recommendations.
Your Role: As a user of an intelligent recommendation system, provide feedback to improve recommendations based on:
1. 【Conversation History】: Previous dialogue between you and the system
2. 【Your Preferences】: Movies and movie types you enjoy
【Conversation History】
{conversation_history}
【Your Preferences】
{liked_genres}
【Feedback Guidelines】:
- Give feedback that **clearly expresses your preferences** based on 【Conversation History】 and 【Your Preferences】.
- You can mention the title of a disliked movie only if necessary, but never reference the title of your preferred movie.
- Your response should be natural, concise, and in one sentence (no more than 80 words), and should **help the system better understand your taste for future recommendations**.

***ChatGPT Input***

I really enjoy **action-packed thrillers** with a strong revenge plot, especially when they star actors like **Keanu Reeves**. A recent film with intense fight scenes and a gripping story about a skilled protagonist taking on the criminal underworld would be perfect. I also appreciate stylish direction and a balance of emotional depth with high-energy action. Something from the last decade with a mix of **crime** and **thriller** elements, produced by a studio known for quality **action** films, would be ideal.

***ChatGPT Output***

Figure 6: prompt feedback

You are a knowledgeable user of an intelligent movie recommendation system, familiar with key movie attributes such as cast, genre, director, and production companies.
Your response should be natural and detailed, and informative, helping the system better understand your preferences and improve future recommendations.
Your task is to provide clear and natural feedback that reveals your preferences based on the 【Conversation History】 and helps the recommendation system refine its recommendation.
Here is the conversation history between you and the recommendation system:
【Conversation History】
{conversation_history}
Here is the recommendation list provided to you by the recommendation system:
【Recommendation List】
{rec_list}
【Task Instructions】
Provide a new response (no more than 150 words) that **reflects your preferences based on the 【Conversation History】**.

***Simulator Feedback Generation Instruction***

Figure 7: Prompt for simulator feedback generation task

You are a movie recommendation expert, skilled at inferring user preferences from conversation history and familiar with key movie attributes such as cast, genre, director, and production companies.
Your task is to analyze the user's preferences based on the 【Conversation History】 between the user and the recommendation system, and identify the movie the user most likely wants to be recommended, along with its key attribute information.
【Conversation History】
{conversation_history}
【Task Instructions】
Your output format should be: **Movie title:** <movie_title>. **Attribute**: {{<attribute_name>: <attribute_values>. <attribute_name>: <attribute_values>. ...}}

***Simulator Attribute Alignment Instruction***

Figure 8: Prompt for simulator attribute alignment Task

You are a knowledgeable user of an intelligent movie recommendation system, familiar with key movie attributes such as cast, genre, director, and production companies.
Your task is to analyze the 【Conversation History】 and determine whether the 【Recommended Movie】 by the recommendation system matches your interests. If does, reply with Yes to indicate acceptance; if not, reply with No to indicate rejection."
【Conversation History】
{conversation_history}
【Recommended Movie】
{rec_movie}
【Task Instructions】
Your output should be strictly one of the following: **Yes or No**.

***Simulator Target Prediction Instruction***

Figure 9: Prompt for simulator target prediction task

You are a professional movie recommendation expert. You are skilled at inferring user preferences from conversation history and are familiar with movie attributes such a cast (actors), genre, director, and production companies.
【Task Instructions】
1. Perform your reasoning strictly according to the following steps within your reasoning process (**between <think> and </think>**):
**Step 1**: Preference Inference
**Step 2**: Attribute Matching
**Step 3**: Scoring
**Step 4**: Ranking and Selection
**Step 5**: Explanation
2. Please output the final recommendation list of 10 movies in the following format:
**RANK LIST: [1. Movie Title (Year), 2. Movie Title (Year), ..., 10. Movie Title (Year)]**
Your task is to recommend the most suitable movies to the user based on the following information through step-by-step reasoning.
Here is the conversation history between you and the user:
【Conversation History】
{conversation_history}
You need to select 10 of the most suitable movies from the following candidate pool and construct a recommendation list in order of priority to recommend to the user:
【Candidate Pool】
{candidate_pool}

***Recommender SFT Instruction***

Figure 10: Prompt for recommender SFT stage training