Towards Effective Structured Context Modeling for Conversational Recommender Systems via Dual-node Monte Carlo Tree Search

Anonymous ACL submission

Abstract

We investigate the role of conversational context modeling in user preference tracking for Conversational Recommendation Systems (CRSs). In this regard, we propose DREAMS, a novel tree-structured context modeling framework that explicitly captures user preference evolution throughout multi-turn interactions. DREAMS introduces two specialized node types to support the two fundamental objectives of CRSs: preference elicitation and preference exploitation. Specifically, elicitation nodes leverage Monte Carlo Tree Search (MCTS) to strategically explore conversational actions and infer latent user preferences, while exploitation nodes employ LLM-based refinement to transform the tracked preference state into structured retrieval queries for recommendation. Extensive experiments on benchmark datasets demonstrate the effectiveness of DREAMS and its design. Our code is available at `https://anonymous.4open.science/r/DREAMS_CRS-F8F2`.

1 Introduction

Conversational Recommendation Systems (CRSs) aim to actively elicit user preferences through multi-turn conversations (i.e., *preference elicitation*) and leverage the acquired preferences to recommend relevant items (*preference exploitation*) (Deng et al., 2021; Lei et al., 2020; He et al., 2023; Deng et al., 2025). Achieving these goals requires accurate user preference tracking (i.e., what the user likes and dislikes), as insufficient tracking can result in redundant questions or premature and irrelevant recommendations, thereby degrading system effectiveness. This, in turn, necessitates robust conversational context modeling (Chen et al., 2025a; Wang et al., 2025), since such user preferences are progressively revealed throughout the conversation. Taking Figure 1 for example, the conversation history may implicitly reveal a negative preference toward a specific director. Such implicit signals should be tracked and consistently suppressed in future recommendations involving that director. Therefore, conversation context modeling forms the foundation of preference tracking, which ultimately supports both preference elicitation and preference exploitation.

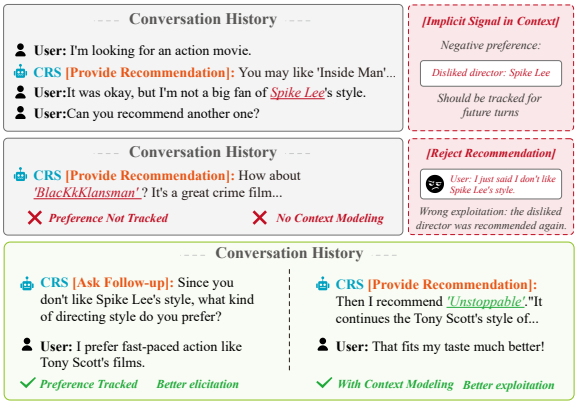

Figure 1: The claim '*not a big fan of Spike Lee's style*' expresses a negative preference that should be retained across turns. Otherwise, the CRS may continue recommending his films, leading to repeated user rejection.

However, existing methods frequently model conversational context as simple free-form text for user preference tracking, which leads to suboptimal performance (Chen et al., 2024; Huang et al., 2024) due to two limitations. 1) For preference elicitation, these methods directly inject the entire conversation history into LLM prompts to decide whether to ask or recommend (Feng et al., 2023; Fang et al., 2024; Wang et al., 2023a; Qin et al., 2024), resulting in information overload and untargeted questioning (Kemper et al., 2024; Zhang et al., 2023). 2) For preference exploitation, they encode free-form conversational context (as a query) into dense embeddings for item retrieval, where irrelevant conversational content introduces noise and degrades retrieval accuracy (Li et al., 2025; Ni et al., 2025; Xu et al., 2025). To ease these limitations, recent studies have represented conversations as JSON-formatted dialogue states (Kemper et al., 2024) or

MCTS (Du et al., 2025; Dao et al., 2024). However, JSON-based methods represent each turn as an isolated state snapshot, overlooking the dependencies among preference states across the conversation. Conversely, MCTS-based methods capture the exploration structure of interactions, but their nodes lack explicit semantic representations of user preferences. What's more, they only partially structures preference elicitation, while not to jointly structure both elicitation and exploitation under a unified framework. As a result, neither paradigm simultaneously enables effective preference tracking.

To validate the above analysis, we conducted a targeted evaluation of existing CRSs in Section 3.1. The results conclusively demonstrate that existing methods conclusively hinder user preference tracking, diminishing both preference elicitation and exploitation, and thus overall system performance. This is evidenced by their difficulty in identifying and utilizing conversation state for accurately timing questions or recommendations. Furthermore, these CRSs exhibit high recommendation error rates, even when the conversation context contains complete user preferences. These findings emphasize the need for advanced context modeling to effectively track user preferences and suggesting that effective user preference tracking is not merely a matter of structuring dialogue history. It requires the structured state to actively drive both preference elicitation and exploitation.

To this end, we propose DREAMS, a **D**ual-node conversational **R**ecomm**E**nd**A**tion system using **M**onte carlo tree **S**earch for effective user preference tracking. In particular, DREAMS explicitly models user preference evolution through a tree-structured conversation state, representing conversation states as Json-structured nodes and state transitions as edges, forming an evolving preference structure throughout the conversation. Importantly, DREAMS introduces two complementary node types. Elicitation Nodes focus on acquiring user preference by strategically exploring alternative conversational actions (e.g., *Director-Inquiry, FailureReflection*) through Monte Carlo Tree Search (MCTS), enabling the system to infer and refine latent user preferences. Exploitation Nodes focus on recommendation by transforming the tracked preference state into structured retrieval queries, thereby reducing contextual noise and improving item retrieval accuracy. As such, DREAMS provides a structured representation of the entire user preference lifecycle, cohesively bridging conversation flow and recommendation queries to optimize the elicitation and exploitation of user information for superior recommendations.

We experimentally demonstrate the effectiveness of DREAMS using the widely-used simulator-based evaluation framework[1] (Wang et al., 2023a; Huang et al., 2024) and two benchmark datasets. The results show that DREAMS achieves superior recommendation accuracy compared to existing methods (+7.43%, on average), exhibiting enhanced performance in both preference elicitation (+9.35%) and exploitation (+4.00%). Further analysis reveals that our structured context modeling not only provides fine-grained and interpretable guidance for conversational planning, but also enables the CRS to accurately associate subtle or implicit contextual signals with the corresponding user preference attributes through structured representations and search over conversational states. In summary, our contributions are as follows:

- We emphasize the importance of structured context modeling for effective preference tracking and systematically analyze the limitations of existing LLM-based CRS methods in this regard.
- We propose DREAMS, a unified framework that jointly models preference elicitation and exploitation through MCTS over structured conversational states.
- Extensive experiments demonstrate the effectiveness of DREAMS and validate the critical role of structured context modeling in conversational recommendation.

2 Related Work

User Preference Tracking. A central challenge in conversational recommendation is maintaining an accurate representation of user preferences as they are progressively revealed throughout multi-turn interactions (Liu et al., 2023c; Deng et al., 2025). Existing LLM-based CRSs primarily support this process from two perspectives. For preference elicitation, prior work focuses on improving dialogue decision-making, such as determining when to ask questions versus provide recommendations (Gao et al., 2023; Friedman et al., 2023; Wang et al., 2023a). For preference exploitation, most methods encode conversational history into semantic representations for downstream retrieval and recommendation (Gao et al., 2023; Liu et al., 2023c; Huang

[1] Human evaluation in Section 5.3 shows our reliability.

et al., 2024; Qin et al., 2024). Our work highlights the importance of structured context modeling for maintaining an evolving preference representation that can simultaneously support both preference elicitation and preference exploitation.

Context Modeling in LLM-based CRS. Most LLM-based CRSs rely on free-form dialogue history as the underlying representation of user preferences, using it for both preference elicitation (Gao et al., 2023; Friedman et al., 2023; Wang et al., 2023a) and preference exploitation (He et al., 2023; Qin et al., 2024). Such unstructured context can obscure preference signals and introduce retrieval noise (Zhang et al., 2023; Kemper et al., 2024; Li et al., 2025; Huang et al., 2024; Ni et al., 2025), even for long-context LLMs[2] (Liu et al., 2023a; Bai et al., 2023; An et al., 2025). Recent work attempts to alleviate this issue from two perspectives. RA-CRS (Kemper et al., 2024) summarizes dialogue history into JSON-formatted preference states, but does not model preference state evolution during conversational exploration. In contrast, MCTS-based CRSs such as SAPIENT (Du et al., 2024) and T-EPL (Dao et al., 2024) explicitly model exploration through tree search, yet their search nodes are not grounded in structured preference states. Consequently, existing methods either provide structured preference states without structured preference evolution, or model exploration trajectories without structured preference states[3]. Motivated by this gap, our framework performs search directly over structured conversational states to jointly support preference elicitation and exploitation.

3 Preliminary Study

We conduct a targeted evaluation to analyze the deficiencies of existing LLM-based CRSs in effective context modeling, leading to challenges in preference elicitation and exploitation. More details are provided in Appendix A.6, A.7.

3.1 Evaluation Setup

Evaluation Overview. Adhering to established practices (Wang et al., 2023a,b; Fang et al., 2024; Qin et al., 2024; Huang et al., 2024), we employ LLM-based user simulators to interact with CRSs. The interaction continues until either the simulator accepted a recommendation or the maximum allowed number of turns is reached. Each resulting conversational history, as the context, is subsequently utilized for evaluation purposes. See Appendix A.9 for details.

Model	Accuracy	Elicitation		Exploitation
	SR↑	FGE^2↓	CGE^2↓	PE^2↓
InterCRS	0.313	0.360	0.570	0.240
MACRS	0.400	0.187	0.456	0.220
RA-CRS	0.333	0.333	0.461	0.240
ChatCRS	**0.440**	**0.120**	0.561	**0.200**
PC-CRS	0.403	0.247	0.491	0.240
SAPIENT-LLM	0.380	0.307	**0.443**	0.233
T-EPL	0.327	0.313	0.469	0.253

Table 1: Elicitation and exploitation error evaluation.

Datasets & User Simulator. Following prior CRS evaluation protocols (Wang et al., 2023a; Huang et al., 2024; Qin et al., 2024), we use LLM-based user simulators with persona-conditioned preference profiles (e.g., genres, actors, and directors). Experiments are conducted on Redial (Li et al., 2018)[4]. More details are provided in Appendix A.9. Complementary human validation is reported in Section 5.2 and Appendix A.8. More details are provided in Appendix A.9.

Baselines. Our preliminary study incorporates free-form-based LLM-based CRSs, including InterCRS(Wang et al., 2023a), MACRS(Fang et al., 2024), ChatCRS(Li et al., 2024), PC-CRS(Qin et al., 2024), and RA-CRS(Kemper et al., 2024). Additionally, we add two MCTS-based baselines, SAPIENT (Du et al., 2025) and T-EPL (Dao et al., 2024). See Appendix A.1 for details.

Evaluation Metrics. We introduce specific error-based metrics to investigate the deficiencies in preference elicitation and exploitation[5]. (1) For preference elicitation, we consider two key metrics: Coarse-Grained Elicitation Error (CGE^2) and Fine-Grained Elicitation Error (FGE^2). The former quantifies the proportion of wrong action selection regarding asking question and making recommendations, while the latter quantifies the proportion of ineffective or misaligned follow-up questions after the system receives specific user feedback about their preferences. (2) To assess preference exploitation, we employ the Preference Exploitation Error (PE^2), which measures the proportion of instances where the CRS fails to retrieve a relevant item for recommenda-

[2]Also evidenced by our experiment in Table 11.
[3]Refer to Appendix A.2 for method comparison.
[4]Additional datasets are introduced in Section 5.1
[5]Detailed in Table 7 in Appendix.

tion despite a fully clarified understanding of user preferences. (3) Furthermore, in accordance with established methodologies (Qin et al., 2024; Fang et al., 2024), we utilize the conventional metrics of Success Rate (SR) to assess the overall recommendation success. Refer to Appendix A.6 for implementation details and Appendix A.8 for reliability analysis regarding LLM-as-judge.

3.2 Experimental Findings

Table 1 shows that off-the-shelf CRS still suffer from persistent elicitation and exploitation errors, highlighting the need for more faithful and fine-grained context modeling. In preference elicitation, coarse-grained action selection remains a major weakness: InterCRS reaches a high CGE^2 of 0.570, indicating difficulty in deciding when to ask questions rather than recommend items. Structured representations alone are insufficient, as RA-CRS still reports high FGE^2 and CGE^2 values of 0.333 and 0.461 despite using JSON-format contexts. Likewise, MCTS-based methods do not guarantee reliable planning, with SAPIENT-LLM and T-EPL obtaining CGE^2 values of 0.443 and 0.469. Fine-grained elicitation is also unreliable, as reflected by InterCRS's FGE^2 of 0.360. For preference exploitation, InterCRS further shows a high PE^2 of 0.240, suggesting that explicitly stated preferences are not effectively converted into retrieval cues. Overall, these results indicate that neither JSON-style representation nor generic MCTS planning is sufficient; CRS requires context modeling that jointly supports state tracking, action selection, and preference-grounded retrieval.

4 DREAMS

Following prior work (Du et al., 2025; Yu et al., 2023), we formulate the CRS interaction as a Markov Decision Process (MDP) $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma \rangle$, where the state space $\mathcal{S}$ represents dialogue context and preference information, and the action space $\mathcal{A}$ contains system actions such as asking clarification questions, reflecting on failed recommendations, and recommending items.

4.1 Overview of DREAMS

DREAMS Formalization. As shown in Figure 2, DREAMS formulates user preference tracking as search over structured conversational states, where nodes encode evolving preference states and edges represent state transitions. Unlike JSON-style trackers such as RA-CRS, which mainly use structured states as turn-independent static memory, DREAMSactively searches over these states for decision-making. It also differs from MCTS-based CRSs such as SAPIENT and T-EPL, whose search nodes are not structured and reusable preference states for open-ended dialogue. Different from prior work that mainly focuses on either elicitation or exploitation, DREAMS contains two node types to jointly optimizing both preference elicitation and exploitation. For preference elicitation, Elicitation Nodes (ELNodes) update the current preference state after each user utterance and perform MCTS over candidate actions. The search process selects promising states, expands actions, simulates future dialogue trajectories, and back-propagates rewards to determine whether the system should ask, reflect, or recommend. For preference exploitation, when MCTS selects `ItemRec`, the corresponding ELNode transitions into an Exploitation Node (EXNode). The EXNode transforms the tracked preference state into structured retrieval queries and selects the query with the best retrieval quality. Therefore, DREAMSperforms retrieval from structured queries derived from evolving preference states rather than directly from noisy dialogue history. Algorithm 1 summarizes the full procedure. We further design a hierarchical and verifiable reward to jointly capture elicitation progress and recommendation success. Detailed formulations are provided in Appendix A.4.

4.2 ELNode for Preference Elicitation

ELNode supports preference elicitation by searching over possible next actions before the system responds. Its action space covers three purposes: preference elicitation, failure reflection, and transition to exploitation[6]. Each ELNode maintains a JSON-style preference state. An LLM parses free-form user utterances into structured key-value updates; for example, "I am not a big fan of Spike Lee's style" is mapped to `<director>: dislike Spike Lee`. The updated state is then used as the basis for MCTS. Here, the core operations — *Selection, Expansion, Simulation, and Backpropagation* — are implemented as follows.

Selection. Given state s_i, DREAMS employs the *Upper Confidence bounds applied to Trees* algorithm[7] (Kocsis and Szepesvári, 2006) to select the

[6]Detailed aciton space in Appendix A

[7]Known for balancing the exploitation (choosing high-

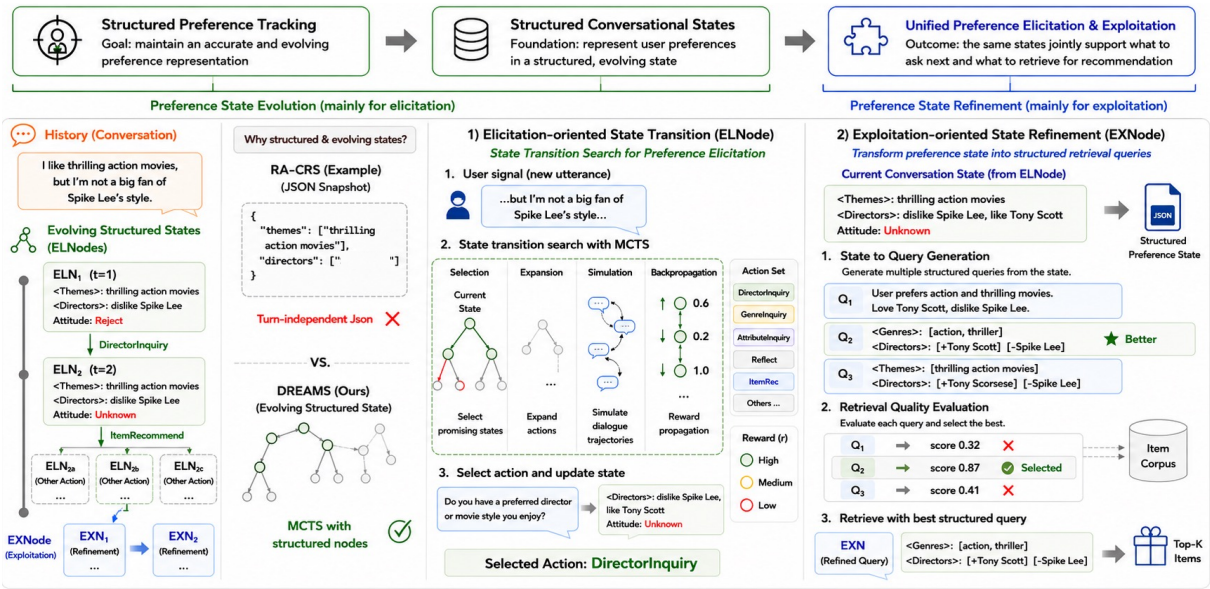

Figure 2: Overview of DREAMS's structured modeling and unified elicitation & exploitation. It incorporates ELNodes to explicitly capture estimated user preferences, and EXNodes to enable structured query refinement.

best action (branch) a^*, where $A(s_i)$ is the set of child nodes of s_i), $R(s_i, a'_i)$ is the total reward accumulated from simulations that passed through all the child nodes of s_i after taking the action a'_i, $N(s_i)$ is the visit count for node s_i, $f(s_i, a_i)$ is the function that returns the child node of s_i after applying action a_i, and c is an exploration constant.

$$\max_{a'_i \in A(s_i)} \left(\frac{R(s_i, a'_i)}{N(f(s_i, a'_i))} + c \cdot \sqrt{\frac{2 \log N(s_i)}{N(f(s_i, a'_i))}} \right) \quad (1)$$

Expansion. Upon reaching a leaf ELNode, the tree expands by attaching new child nodes representing candidate elicitation actions. To efficiently prune the search space and prioritize promising actions, we leverage an LLM to generate prior probabilities $\mathcal{M}$ over the available actions. This LLM guidance ensures that only actions with a positive predicted score are expanded. To reduce the variance of a single LLM call, we sample the LLM m times and aggregate the resulting priors.

Simulation. To estimate the long-term effectiveness of an expanded action, DREAMS rolls out the dialogue based on a probability-guided policy provided by the LLM until a terminal state or a predefined depth limit is reached. The simulator is initialized with the preference state stored in the current ELNode, and subsequent actions are sampled from the LLM-guided policy. Each transition receives the hierarchical reward described in Appendix A.4, discounted by γ.

value nodes) and exploration (choosing less-visited nodes).

Back-propagation. After each rollout, DREAMS back-propagates the discounted cumulative reward to all nodes on the simulated path and updates their visit counts and expected rewards:

$$R(s_i, a_i) = \frac{1}{K} \sum_{j=1}^{K} \left(\sum_{s' \in S^j, a' \in A^j} \gamma \cdot r(s', a') \right) \quad (2)$$

The reward $r(\cdot)$ consists of three parts: an attitude reward for successful recommendations, an information acquisition reward for learning new preference details, and a turn penalty for conversation efficiency[8]. After a fixed number of MCTS iterations, DREAMS selects the final action by combining the tree value with the LLM prior:

$$\max_{a} \left(W_{tree} \cdot \frac{N(f(s_0, a))}{\sum N(f(s_0))} + W_{LLM} \cdot \mathcal{M}(s_0, a) \right) \quad (3)$$

where s_0 is the root state, and W_{tree} and W_{LLM} control the contributions of search statistics and LLM priors.

4.3 EXNode for Preference Exploitation

When `ItemRec` isselected, it expands an EXNode to convert the tracked preference state into a retrieval-friendly query with following two steps:

Refinement Mechanism. The EXNode takes the current dialogue history, the JSON-style preference state, and the raw user query as input. It first summarizes the cumulative preference state s_t^R, which differs from the ELNode state s_t: s_t focuses on

[8] See Appendix A.4 for details

Model	Redial (Movie)				OpendialKG (Movie)				OpendialKG (Book)			
	R@1(↑)	R@5(↑)	R@10(↑)	SR(↑)	R@1(↑)	R@5(↑)	R@10(↑)	SR(↑)	R@1(↑)	R@5(↑)	R@10(↑)	SR(↑)
PLM-based												
BARCOR (Wang et al., 2022a)	0.193	0.467	0.593	0.107	0.094	0.383	0.494	0.211	0.106	0.375	0.456	0.219
UniCRS (Wang et al., 2022b)	0.147	0.373	0.520	0.133	0.233	0.428	0.583	0.250	0.225	0.406	0.563	0.231
LLM-based												
InterCRS (Wang et al., 2023a)	0.213	0.370	0.500	0.313	0.439	0.650	0.783	0.494	0.356	0.469	0.594	0.394
MACRS (Fang et al., 2024)	0.310	0.410	0.560	0.400	0.506	0.672	0.800	0.522	0.506	0.638	0.681	0.544
ChatCRS (Li et al., 2024)	0.367	0.473	0.573	0.440	0.539	0.650	0.772	0.550	0.494	0.600	0.675	0.531
PC-CRS (Qin et al., 2024)	0.307	0.427	0.493	0.407	0.556	0.656	0.772	0.567	0.519	0.625	0.694	0.563
RA-CRS (Kemper et al., 2024)	0.273	0.340	0.387	0.333	0.422	0.606	0.733	0.433	0.494	0.575	0.644	0.519
SAPIENT-LLM (Du et al., 2025)	0.287	0.387	0.427	0.380	0.433	0.644	0.661	0.450	0.488	0.538	0.585	0.506
T-EPL (Dao et al., 2024)	0.293	0.367	0.387	0.327	0.416	0.611	0.639	0.427	0.475	0.550	0.619	0.494
DREAMS (*Ours*)	**0.507**	**0.633**	**0.747**	**0.560**	**0.600**	**0.728**	**0.844**	**0.639**	**0.550**	**0.656**	**0.719**	**0.594**

Table 2: Overall performance of different methods.

the latest turn-level update, while s_t^R aggregates user preferences across the whole dialogue. Given s_t^R, the LLM proposes refinement actions a_i^R, such as removing redundant context, reordering preference attributes, and converting informal expressions into machine-readable constraints. For example, a negative preference such as "dislike Spike Lee's style" can be converted into `<director>: dislike Spike Lee`. Applying the i-th refinement action produces a refined query state:

$$EXN_1^i \leftarrow a_i^R(EXN_0) \tag{4}$$

Evaluation and Selection. DREAMS then retrieves items with each refined query and scores the result by matching the top retrieved items against the preferred attributes in s_t^R:

$$R^R(EXN_1^i) = \text{score}(\text{retrieve}(EXN_1^i)) \tag{5}$$

The query with the highest retrieval score is used for final recommendation generation. In this way, EXNode connects preference tracking to preference exploitation: the system doesn't retrieve from noisy dialogue history directly, but from a structured query derived from evolving preference state.

5 Experiments

5.1 Experimental Setup

Baselines. Beyond LLM-based CRSs in Section 3.1, we follow previous studies (Qin et al., 2024; Wang et al., 2023a) and involve two pre-trained language model based CRSs (BARCOR (Wang et al., 2022a) and UniCRS (Wang et al., 2022b)) and two MCTS-based CRS baselines (SAPIENT (Du et al., 2025) and T-EPL (Dao et al., 2024)). See Appendix A.1 for details.

Datasets & User Simulator. In addition to the datasets and user simulators in Section 3.1, we follow current common practice in the field (Qin et al., 2024; Huang et al., 2024; Chen et al., 2025b) and incorporate OpendialKG (Moon et al., 2019), a multi-domain benchmark dataset.

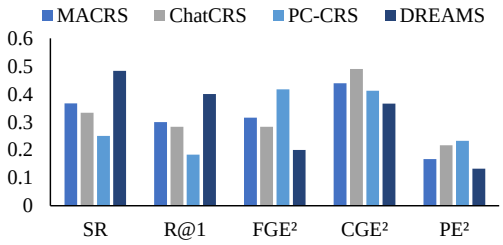

Figure 3: Results on human interaction and evaluation.

Evaluation Metrics. In addition to the metrics in Section 3.1, we incorporate a human evaluation, which leverages human judgment to further assess the performance of CRSs, specifically focusing on preference elicitation and exploitation.

5.2 Main Results

DREAMS consistently outperforms existing CRS baselines. Table 2 shows that DREAMS achieves the best performance across all datasets and metrics. Relative to ChatCRS, the strongest non-MCTS baseline, DREAMS improves average R@1 and SR by 8.57% and 9.07%, respectively. Furthermore, it substantially surpasses the MCTS-based methods SAPIENT-LLM and T-EPL in SR. This suggests that our improvement stems not merely from search, but from grounding search in structured preference states and retrieval refinement.

DREAMS exhibits superior performance in preference state tracking, demonstrating enhanced both elicitation and exploitation of user preferences. As shown in Table 3, DREAMS consistently yields lower elicitation and exploitation errors than all baselines, reflected by its superior CGE^2, FGE^2, and PE^2 scores. The lower CGE^2 indicates fewer coarse-grained decision errors in whether to continue elicitation or proceed to recommendation, while the lower FGE^2 demonstrates stronger fine-grained preference modeling. These gains further

Model	Redial (Movie)				OpendialKG (Movie)				OpendialKG (Book)			
	SR↑	FGE^2↓	CGE^2↓	PE^2↓	SR↑	FGE^2↓	CGE^2↓	PE^2↓	SR↑	FGE^2↓	CGE^2↓	PE^2↓
MACRS (Fang et al., 2024)	0.400	0.187	0.456	0.220	0.522	0.150	0.431	0.106	0.544	0.156	0.498	0.088
ChatCRS (Li et al., 2024)	0.440	0.120	0.561	0.200	0.550	0.133	**0.291**	0.117	0.531	0.169	0.455	0.100
RA-CRS (Kemper et al., 2024)	0.333	0.333	0.461	0.240	0.433	0.161	0.398	0.122	0.519	0.188	0.469	0.106
SAPIENT-LLM (Du et al., 2025)	0.380	0.307	0.443	0.233	0.450	0.150	0.348	0.139	0.506	0.175	0.458	0.119
T-EPL (Dao et al., 2024)	0.327	0.313	0.469	0.253	0.427	0.144	0.332	0.133	0.494	0.194	0.471	0.131
DREAMS	**0.560**	**0.100**	**0.289**	**0.160**	**0.639**	**0.117**	0.304	**0.078**	**0.594**	**0.138**	**0.408**	**0.069**

Table 3: Elicitation and exploitation evaluation of DREAMS and leading baselines.

lead to lower exploitation errors (PE^2), suggesting that the inferred preference state can be more effectively transformed into retrieval queries. A more specific conclusion is that DREAMS does not benefit from MCTS in isolation. Instead, its gains come from using tree search over a structured preference state, so the system can decide when to clarify, when to recommend, and how to refine the retrieval query from the tracked preferences.

DREAMS enjoys enhanced practical utility in real-world human interactions. To evaluate practical utility, we conduct a human study involving 60 participants, who interact with DREAMS and three representative baselines (MACRS, ChatCRS, and PC-CRS) under predefined preference settings. As shown in Figure 3, DREAMS consistently achieves the best performance across all metrics, including SR, Recall@1, CGE^2, FGE^2, and PE^2, indicating superior preference elicitation, preference exploitation, and recommendation effectiveness in realistic conversational settings. To validate the reliability of our automatic evaluator, we randomly sample 50 dialogues generated by DREAMS on Redial and ask 10 human annotators to label PE^2, FGE^2, and CGE^2 following the same criteria. The resulting Krippendorff's α scores are 0.74 and 0.68 for Elicitation Error and Exploitation Error, respectively, indicating substantial agreement and supporting the reliability of the automatic evaluation, consistent with prior findings (Liu et al., 2023b).

DREAMS generalizes across LLM backbones. Table 11 shows that DREAMS maintains clear advantages on both Gemini-2.5-Flash and GPT-4o-mini. Despite Gemini's strong long-context capability, the free-form method InterCRS still suffers from high elicitation errors, whereas DREAMS consistently achieves better preference tracking and recommendation performance. These results suggest that the limitation of free-form context modeling is not primarily due to context length. Instead, effective conversational recommendation requires an explicit preference state that supports preference elicitation and exploitation.

5.3 Recommendation Performance Analysis

We analyze the contribution of different components through the following variants:

- w/o EXNode. Directly retrieves from free-form dialogue history without query refinement.
- w/o ELNode. Replaces MCTS-based state transition with direct LLM action selection.
- w/o ELNode & Json. Further removes structured preference states and tracks preferences using free-form context.
- w/ Json Only. Uses JSON-style preference states, but performs action selection and retrieval without structured search.

Table 4 disentangles the effects of *structured preference states*, *state-transition search*, and *structured retrieval refinement*. Removing both ELNode and JSON causes the largest degradation, significantly increasing $FGE^2$and CGE^2, showing that explicit structured states are essential for effective preference tracking. Removing ELNode while preserving JSON also degrades performance, indicating that preference states must be actively updated and utilized during interaction. Replacing MCTS with direct LLM reasoning further reduces SR and increases FGE^2, suggesting that structured states should serve as actionable decision states rather than passive memory. Removing EXNode mainly decreases Recall@1 and SR while increasing PE^2, demonstrating the importance of exploitation-side query refinement. Overall, the results show that structured preference states, state-transition search, and structured retrieval refinement are all critical to unified preference elicitation and exploitation.

5.4 Preference Elicitation Analysis

Structured conversational states improve preference elicitation by transforming noisy dialogue history into stable and decision-relevant preference representations. Table 4 supports this conclusion through the ELNode and JSON-related ablations. Removing ELNode weakens the model's ability to maintain evolving preference states, increasing both $FGE^2$and CGE^2. Further removing

Model	Redial (Movie)					OpendialKG (Movie)					OpendialKG (Book)				
	R@1(↑)	SR(↑)	FGE^2 (↓)	CGE^2(↓)	PE^2 (↓)	R@1(↑)	SR(↑)	FGE^2 (↓)	CGE^2(↓)	PE^2 (↓)	R@1(↑)	SR(↑)	FGE^2 (↓)	CGE^2(↓)	PE^2 (↓)
DREAMS	**0.507**	**0.560**	**0.100**	**0.289**	**0.160**	**0.600**	**0.639**	**0.117**	**0.304**	**0.078**	**0.550**	**0.594**	**0.138**	**0.408**	0.069
Structured Context Modeling															
w/o EXNode	0.407	0.487	0.133	0.311	0.193	0.517	0.572	0.139	0.326	0.117	0.494	0.556	0.144	0.422	0.100
w/o ELNode	0.393	0.427	0.173	0.417	0.173	0.500	0.550	0.156	0.378	0.094	0.444	0.456	0.188	0.490	**0.063**
w/o ELNode & Json	0.287	0.347	0.340	0.539	0.233	0.450	0.478	0.406	0.455	0.111	0.388	0.456	0.250	0.523	0.075
Structured Tracking															
w/ Json Only	0.327	0.360	0.200	0.455	0.240	0.439	0.472	0.411	0.384	0.128	0.394	0.475	0.213	0.473	0.081

Table 4: Ablation studies on benchmark datasets.

JSON-style states leads to even larger degradation. These results show that explicit structured state modeling is essential for effective preference elicitation, enabling DREAMS to outperform MCTS-based baselines such as SAPIENT-LLM and T-EPL. Case study (See Appendix B.2) illustrates how DREAMS differs from baselines facing noisy dialogue history.

JSON-format state tracking records preferences explicitly, but structured search is needed to turn the recorded state into better decision making. Both RA-CRS and the w/Json Only variant maintain structured preference states, yet neither operationalizes them through structured search. Table 4 isolates this effect: although the w/Json Only variant preserves the same structured states and action/query interfaces, replacing MCTS with direct LLM reasoning substantially lowers SR and increases FGE^2. This demonstrates that effective preference tracking requires not only structured states, but also search over evolving conversational states for lookahead decision-making. A mechanism-level case study is provided in Appendix B.2.

5.5 Preference Exploitation Analysis

By leveraging the refinement mechanism, the EXNode progressively refines the retrieval queries, steering them closer to the ground-truth item description. As shown in Figure 4, EXNode progressively refines retrieval queries toward the ground-truth item description, yielding consistently higher cosine similarity than original queries. In contrast, randomly paraphrased LLM queries are unstable and can even underperform the original query, highlighting the sensitivity of retrieval to contextual noise (Huang et al., 2024). We further observe that EXNode tends to generate more structured JSON-style retrieval queries, which better align with the structured knowledge bases in REDIAL and OpenDialKG. Therefore, EXNode improves exploitation through both semantic refinement and structured retrieval representations.

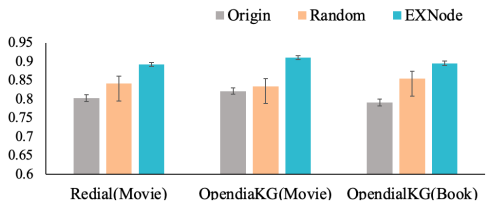

Figure 4: Comparison of query refinement methods via cosine similarity to ground-truth embeddings.

5.6 Efficiency Analysis

Although a common concern with tree-based planning is latency, Table 10 shows that vanilla DREAMS already achieves efficiency comparable to existing CRSs through our parallel function calling adaptation. To further improve deployment efficiency, we introduce **DREAMS(EA)**, an Experience-Augmented variant inspired by prior experience-based reasoning methods (Gu et al., 2025; Ren et al., 2025). Specifically, we store MCTS trajectories as structured experience knowledge and retrieve structurally similar successful/-failed cases from an Experience Knowledge Base (EKB) to warm-start decision making, thereby bypassing expensive online tree search. As shown in Table 10, DREAMS(EA) reduces inference latency to 9s while maintaining competitive recommendation performance (Recall@1=0.467). More implementation and efficiency details are provided in Appendix B.1.

6 Conclusion

In this paper, we highlight the critical role of structured context modeling in conversational recommender systems and propose DREAMS, a dual-node MCTS framework that jointly optimizes preference elicitation and preference exploitation. Extensive experiments demonstrate that effective conversational recommendation requires not only structured representations, but also structured search over evolving conversational states. We hope this work encourages future CRS research to treat structured context as an important active decision mechanism.

Limitations

To ensure a fair comparison, we adhered to established protocols (Qin et al., 2024; Li et al., 2024) by limiting all methods to the GPT-4o-mini model as their backbone. Due to constraints in computational resources and budget, we do not extend our evaluation to other SOTA large language models (e.g., Claude Opus). This limited model selection may introduce potential bias in the field of LLM-based CRS research, as different models employ distinct fine-tuning strategies and alignment mechanisms that could impact their performance, particularly in preference elicitation. We encourage future research to investigate the influence of diverse LLM backbones on CRS performance.

More broadly, the evaluation of modern CRSs remains challenging because scalable protocols often depend on LLM-based judges, either as offline evaluators or as user simulators. Although such protocols enable controlled and reproducible comparisons, they may inherit biases from the evaluator model, prompt design, and predefined user preference ontology. Future work should develop more reliable human-grounded benchmarks, behaviorally validated user simulators, and evaluator-independent protocols for conversational recommendation.

References

Guojia An, Jie Zou, Jiwei Wei, Chaoning Zhang, Fuming Sun, and Yang Yang. 2025. Beyond whole dialogue modeling: Contextual disentanglement for conversational recommendation. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 31–41.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. 2023. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*.

Bo Chen, Xinyi Dai, Huifeng Guo, Wei Guo, Weiwen Liu, Yong Liu, Jiarui Qin, Ruiming Tang, Yichao Wang, Chuhan Wu, et al. 2024. All roads lead to rome: Unveiling the trajectory of recommender systems across the llm era. *arXiv preprint arXiv:2407.10081*.

Luyu Chen, Quanyu Dai, Zeyu Zhang, Xueyang Feng, Mingyu Zhang, Pengcheng Tang, Xu Chen, Yue Zhu, and Zhenhua Dong. 2025a. Recusersim: A realistic and diverse user simulator for evaluating conversational recommender systems. In *Companion Proceedings of the ACM on Web Conference 2025*, pages 133–142.

Nuo Chen, Quanyu Dai, Xiaoyu Dong, Xiao-Ming Wu, and Zhenhua Dong. 2025b. Evaluating conversational recommender systems with large language models: A user-centric evaluation framework. *arXiv preprint arXiv:2501.09493*.

Huy Quang Dao, Yang Deng, Khanh-Huyen Bui, Dung D. Le, and Lizi Liao. 2024. Experience as source for anticipation and planning: Experiential policy learning for target-driven recommendation dialogues. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 14179–14198, Miami, Florida, USA. Association for Computational Linguistics.

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.

Yang Deng, Lizi Liao, Wenqiang Lei, Grace Hui Yang, Wai Lam, and Tat-Seng Chua. 2025. Proactive conversational ai: A comprehensive survey of advancements and opportunities. *ACM Transactions on Information Systems*, 43(3):1–45.

Hanwen Du, Bo Peng, and Xia Ning. 2024. Sapient: Mastering multi-turn conversational recommendation with strategic planning and monte carlo tree search. *arXiv preprint arXiv:2410.09580*.

Hanwen Du, Bo Peng, and Xia Ning. 2025. SAPIENT: Mastering multi-turn conversational recommendation with strategic planning and Monte Carlo tree search. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2629–2648, Albuquerque, New Mexico. Association for Computational Linguistics.

Jiabao Fang, Shen Gao, Pengjie Ren, Xiuying Chen, Suzan Verberne, and Zhaochun Ren. 2024. A multi-agent conversational recommender system. *arXiv preprint arXiv:2402.01135*.

Yue Feng, Shuchang Liu, Zhenghai Xue, Qingpeng Cai, Lantao Hu, Peng Jiang, Kun Gai, and Fei Sun. 2023. A large language model enhanced conversational recommender system. *arXiv preprint arXiv:2308.06212*.

Luke Friedman, Sameer Ahuja, David Allen, Zhenning Tan, Hakim Sidahmed, Changbo Long, Jun Xie, Gabriel Schubiner, Ajay Patel, Harsh Lara, et al. 2023. Leveraging large language models in conversational recommender systems. *arXiv preprint arXiv:2305.07961*.

Yunfan Gao, Tao Sheng, Youlin Xiang, Yun Xiong, Haofen Wang, and Jiawei Zhang. 2023. Chatrec: Towards interactive and explainable llms-augmented recommender system. *arXiv preprint arXiv:2303.14524*.

Jiawei Gu, Ziting Xian, Yuanzhen Xie, Ye Liu, Enjie Liu, Ruichao Zhong, Mochi Gao, Yunzhi Tan, Bo Hu, and Zang Li. 2025. Toward structured knowledge reasoning: Contrastive retrieval-augmented generation on experience. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 23891–23910, Vienna, Austria. Association for Computational Linguistics.

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.

Chen Huang, Peixin Qin, Yang Deng, Wenqiang Lei, Jiancheng Lv, and Tat-Seng Chua. 2024. Concept–an evaluation protocol on conversation recommender systems with system-and user-centric factors. *arXiv preprint arXiv:2404.03304*.

Sara Kemper, Justin Cui, Kai Dicarlantonio, Kathy Lin, Danjie Tang, Anton Korikov, and Scott Sanner. 2024. Retrieval-augmented conversational recommendation with prompt-based semi-structured natural language state tracking. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2786–2790.

Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer.

Wenqiang Lei, Xiangnan He, Yisong Miao, Qingyun Wu, Richang Hong, Min-Yen Kan, and Tat-Seng Chua. 2020. Estimation-action-reflection: Towards deep interaction between conversational and recommender systems. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 304–312.

Chuang Li, Yang Deng, Hengchang Hu, Min-Yen Kan, and Haizhou Li. 2024. Incorporating external knowledge and goal guidance for llm-based conversational recommender systems. *arXiv preprint arXiv:2405.01868*.

Qianlong Li, Chen Huang, Shuai Li, Yuanxin Xiang, Deng Xiong, and Wenqiang Lei. 2025. GraphOTTER: Evolving LLM-based graph reasoning for complex table question answering. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 5486–5506, Abu Dhabi, UAE. Association for Computational Linguistics.

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.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023a. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023b. G-eval: Nlg evaluation using gpt-4 with better human alignment. *arXiv preprint arXiv:2303.16634*.

Yuanxing Liu, Wei-Nan Zhang, Yifan Chen, Yuchi Zhang, Haopeng Bai, Fan Feng, Hengbin Cui, Yongbin Li, and Wanxiang Che. 2023c. Conversational recommender system and large language model are made for each other in e-commerce pre-sales dialogue. *arXiv preprint arXiv:2310.14626*.

Seungwhan Moon, Pararth Shah, Anuj Kumar, and Rajen Subba. 2019. Opendialkg: Explainable conversational reasoning with attention-based walks over knowledge graphs. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 845–854.

Bo Ni, Zheyuan Liu, Leyao Wang, Yongjia Lei, Yuying Zhao, Xueqi Cheng, Qingkai Zeng, Luna Dong, Yinglong Xia, Krishnaram Kenthapadi, Ryan Rossi, Franck Dernoncourt, Md Mehrab Tanjim, Nesreen Ahmed, Xiaorui Liu, Wenqi Fan, Erik Blasch, Yu Wang, Meng Jiang, and Tyler Derr. 2025. Towards trustworthy retrieval augmented generation for large language models: A survey.

Peixin Qin, Chen Huang, Yang Deng, Wenqiang Lei, and Tat-Seng Chua. 2024. Beyond persuasion: Towards conversational recommender system with credible explanations. *arXiv preprint arXiv:2409.14399*.

Yanwei Ren, Haotian Zhang, Fuxiang Wu, Jiayan Qiu, Jiaxing Huang, Baosheng Yu, and Liu Liu. 2025. Sigma: Refining large language model reasoning via sibling-guided monte carlo augmentation.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. 2024. Large language models are not fair evaluators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9440–9450, Bangkok, Thailand. Association for Computational Linguistics.

Ting-Chun Wang, Shang-Yu Su, and Yun-Nung Chen. 2022a. Barcor: Towards a unified framework for conversational recommendation systems. *arXiv preprint arXiv:2203.14257*.

Xiaolei Wang, Xinyu Tang, Wayne Xin Zhao, Jingyuan Wang, and Ji-Rong Wen. 2023a. 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*, SIGIR '25, page 75–84, New York, NY, USA. Association for Computing Machinery.

Xiaolei Wang, Kun Zhou, Xinyu Tang, Wayne Xin Zhao, Fan Pan, Zhao Cao, and Ji-Rong Wen. 2023b. Improving conversational recommendation systems via counterfactual data simulation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2398–2408.

Xiaolei Wang, Kun Zhou, Ji-Rong Wen, and Wayne Xin Zhao. 2022b. 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.

Haozhe Xu, Xiaohua Wang, Changze Lv, and Xiaoqing Zheng. 2025. Beyond single labels: Improving conversational recommendation through LLM-powered data augmentation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15573–15590, Vienna, Austria. Association for Computational Linguistics.

Xiao Yu, Maximillian Chen, and Zhou Yu. 2023. Prompt-based monte-carlo tree search for goal-oriented dialogue policy planning.

Gangyi Zhang, Chongming Gao, Wenqiang Lei, Xiaojie Guo, Shijun Li, Hongshen Chen, Zhuozhi Ding, Sulong Xu, and Lingfei Wu. 2023. Vague preference policy learning for conversational recommendation. *arXiv preprint arXiv:2306.04487*.

Tong Zhang, Chen Huang, Yang Deng, Hongru Liang, Jia Liu, Zujie Wen, Wenqiang Lei, and Tat-Seng Chua. 2024. Strength lies in differences! improving strategy planning for non-collaborative dialogues via diversified user simulation. *arXiv preprint arXiv:2403.06769*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

A Implementation Details

All experiments are conducted using a single Nvidia RTX A6000 GPU.

A.1 Implementation of baselines

Baseline implementations are sourced from their respective official code repositories on GitHub or publicly available checkpoints. GPT-4o-mini is employed as the LLM backbone and *text-embedding-ada-002* is used for recommendation module for all the LLM-based CRSs.

Notably, We adapt the two MCTS-based CRS baselines(SAPIENT(Du et al., 2025) and T-EPL(Dao et al., 2024)) to our dialogue-based LLM-powered setting through a unified high-level planning interface.

- **SAPIENT-LLM.** Consider that SAPIENT is originally designed for attribute-based CRS with discrete ask/recommend actions over attribute values and items. To adapt it to our dialogue-based LLM-powered CRS setting, we replace the original attribute-value state with an LLM-maintained dialogue state, including the conversation history, extracted positive and negative user preferences, rejected items, and the current candidate item set. We preserve its high-level action space, where ask selects a preference facet inferred from item metadata and dialogue context, and recommend selects items from the current candidate ranking. The selected abstract action is then verbalized by the LLM into a natural-language response, while the subsequent user utterance is parsed by the LLM state tracker to update the dialogue state.

- **T-EPL.** Consider that T-EPL is originally formulated for target-driven recommendation dialogues and assumes access to a predefined target item. To make it comparable in our dialogue-based CRS setting, we remove access to the ground-truth target item and replace its target-conditioned value function with a belief-weighted value over the current candidate distribution. Specifically, the base recommender produces a ranked candidate set given the current dialogue state, and T-EPL estimates the value of a simulated state by retrieving similar past dialogue states from memory and aggregating their outcome scores weighted by the current candidate probabilities. During MCTS expansion, candidate actions are represented as abstract dialogue acts, such as asking about an uncertain preference, recommending an item, clarifying ambiguous constraints, or justifying a recommendation, and are finally realized as natural-language responses by the LLM.

A.2 Difference between DREAMS and baselines

As shown in Table 5

A.3 Implementation of DREAMS

In this section, we will introduce in detail the implementation of different components in DREAMS , including: actions for ELNode transition, state update component, LLM scorer, online simulator, reward function and EXNode refinement.

A.3.1 Actions for ELNode transition

DREAMS employs an iterative Monte Carlo search algorithm to identify the optimal action at each round from the action set defined in Table 6. Among the available actions, three are dedicated for preference **inquiry**, corresponding to three key user attributes: genre, actor, and director. These actions elicit user preferences by generating targeted utterances.

FailureReflection serves a special role. Triggered when the user provides negative feedback, it reflects upon the current dialogue context and the received feedback to infer the most appropriate subsequent action from the action set.

The **ItemRecommendation** action is central to the system, as it initiates the use of the refining-chain. This action involves several intermediate steps, including query refinement, retrieval, and reranking. During query refinement, the refining-chain is utilized to generate an optimized query, which is then used to retrieve candidate items. Subsequently, a large language model (LLM) reranks the retrieved candidates and selects the most suitable item for recommendation.

Lastly, the **ItemExplanation** action is designed to provide users with a detailed explanation of the recommended item's plot, thereby increasing user acceptance of the recommendation.

The specific implementation prompts associated with each action are provided in Appendix C.

A.3.2 State Update Component

In order to dynamically and adaptively model user preferences, DREAMS uses a large model to structure free-form text to achieve continuous updating

Model	Elicitation	Exploitation	Design Highlights	Task
SAPIENT	MCTS with free-form node	Free-form (whole dialogue)	Free-form node MCTS just for elicitation.	Attr.
T-EPL	MCTS with free-form node	Free-form (whole dialogue)	Free-form node MCTS just for topic planning.	Target
InterCRS	Free-form (whole dialogue)	Free-form (whole dialogue)	LLMs' performance evaluation.	Dialog
MACRS	Free-form (whole dialogue)	Free-form (whole dialogue)	Agents collaboration.	Dialog
PC-CRS	Free-form (whole dialogue)	Free-form (whole dialogue)	Persuasiveness and credibility study.	Dialog
ChatCRS	Free-form (whole dialogue)	Free-form (whole dialogue)	Domain-specific knowledge enhancement.	Dialog
RA-CRS	Turn-independent JSON	Free-form (whole dialogue)	JSON for preference elicitation.	Dialog
DREAMS	MCTS with structured node	MCTS with structured node	MCTS for both elicitation and exploitation with structured node	Dialog

Table 5: Comparison of conversational recommendation methods from the perspective of preference elicitation and exploitation. Existing approaches primarily rely on free-form dialogue context or partially structured elicitation, whereas DREAMSjointly models elicitation and exploitation through MCTS over structured conversational states.

Action	Intermediate Step	Return
GenreInquiry	None	utterance
StarInquiry	None	utterance
DirectorInquiry	None	utterance
FailureReflection	None	*function*
ItemRecommendation	*QueryRefine + Retrieval + Rerank*	utterance
ItemExplanation	None	utterance

Table 6: Action sets of p-tree.

of the conversation state. See Figure **??** for details of the state structure and prompts.

A.3.3 LLM Scorer

The LLM scorer is used for providing prior probabilities over possible actions for the tree node expansion and the optimal action selection after iterations. To enhance robustness, we sample the LLM's predictions m times and aggregate the results into a probability distribution. See Figure 8 for details and prompts of each prediction time.

A.3.4 Online Simulator

During simulation of planning-tree, we conduct an online user simulator to give feedback in response to simulated actions. See detail prompts in Figure **??**

A.3.5 Pseudo-Code

As shown in Algorithm 1

Algorithm 1: Pseudo-code of DREAMS

Input : Iteration Steps $n(El)$ and $n(Ex)$, Simulation Depth $k(El)$, Max Turn T
Output : CRS Response

Set $ELNode_0 = \{\}$, Dialogue History $H = \{\}, t = 1$
while *dialogue turn* $t < T$ **do**
  Receive User Utterance u_t
  Update $ELNode_t$ using $ELNode_{t-1}$ and u_t
  `/* 1. ELNode Process Start */`
  for $i \leftarrow 1$ **to** $n(El)$ **do**
    Perform MCTS with max tree depth being $k(El)$, and obtain the estimated value for each action
  Obtain best action a^* based on multiple MCTS trials
  `/* 2. EXNode Process Start */`
  if $a^* =$ *ItemRecommendation* **then**
    Initialize query $EXNode_0 = H$
    LLM generates refinement actions $[a_1^R, ...a_j^R, ..., a_{n(Ex)}^R]$
    Obtain query scores $[R_1^R, ...R_j^R, ..., R_{n(Ex)}^R]$ using Eq.5
    Recommend Item I using query with highest score
    if *User Accepts* I **then**
      Return Successful Recommendation
  else
    Generate Response r_t aligning with a^*
  $H \leftarrow H \cup \{u_t, r_t\}, t = t + 1$
Return Dialogue Termination (Max Turns Reached)

A.4 Reward Design

The reward in DREAMS is designed as a hierarchical combination of intermediate and final signals, rather than a single sparse success indicator. For a simulated transition (s_t, a_t, s_{t+1}), we define the immediate reward as:

$$r_t = \lambda_e r_t^{\text{explore}} + \lambda_r r_t^{\text{retrieve}} + \lambda_u r_t^{\text{attitude}} - \lambda_c r_t^{\text{cost}}, \tag{6}$$

where r_t^{explore} measures whether the current action improves the completeness of the structured preference state, r_t^{retrieve} measures whether the refined query leads to retrieval results that are more consistent with the hidden target profile, r_t^{attitude} captures the user simulator's feedback after interaction, and r_t^{cost} penalizes unnecessarily long or repetitive tra-

jectories.

The overall action value is computed as the discounted return:

$$R(s_t, a_t) = \mathbb{E}\left[\sum_{k=t}^{T} \gamma^{k-t} r_k + \lambda_f \cdot \mathrm{Accept}_T\right], \tag{7}$$

where $\mathrm{Accept}_T \in \{0, 1\}$ indicates whether the simulated dialogue ends with user acceptance.

For exploration, r_t^{explore} is positive when the action helps clarify previously missing or low-confidence attributes in the structured user state. This encourages the planner to ask informative questions when the current preference state is still incomplete.

For exploitation, r_t^{retrieve} is defined using verifiable signals from the environment. Concretely, we compare the attributes of the top retrieved item with the ground-truth attribute profile underlying the simulator. Let $\hat{y}_t^{(1)}$ denote the top-1 retrieved item and let $A(\cdot)$ denote its attribute set. We compute retrieval reward from the overlap between $A(\hat{y}_t^{(1)})$ and the target attribute set, so that higher reward is assigned only when retrieval becomes more faithful to the actual user preference profile. This makes the retrieval reward externally checkable, instead of relying solely on free-form LLM scoring.

Finally, r_t^{attitude} rewards trajectories that lead to explicit user acceptance and penalizes rejection, while r_t^{cost} discourages overly long dialogues. In this way, the reward design combines dense intermediate supervision with a final outcome objective. Although DREAMS does not optimize the model with reinforcement learning, this design is conceptually aligned with recent verifiable-reward based paradigms, in that the key reward terms are grounded in observable and reproducible signals from the environment.

A.4.1 R-chain Refinement

The R-chain first processes the multi-turn dialogue history to extract explicit user preferences regarding genres, actors, directors, and thematic elements (See Figure **??** for detailed prompts). This extraction is performed by prompting an LLM with a structured instruction that emphasizes fine-grained disambiguation of closely related genres (e.g., "thriller" vs. "crime") and returns a structured JSON of liked/disliked entities with associated confidence scores.

Once preferences are extracted, the system formulates candidate retrieval queries using strategies defined by the LLM. A lightweight Monte Carlo Tree is employed to select the optimal refinement strategy. Specifically, a tree of strategy nodes is constructed, where each node corresponds to a paraphrasing strategy and is evaluated based on the simulated performance of the resulting query. During simulation, a query is generated via the LLM based on the node's strategy and then evaluated using a two-part reward function: (1) a preference match score that quantifies how well top retrieved items align with the user's stated preferences, and (2) a potential attitude score that estimates user acceptance likelihood based on cumulative preference overlap and diversity of retrieved genres. The combined reward guides backpropagation in the tree, leading to the selection of the most promising strategy.

The final refined query (See Figure **??** for details) is then embedded and used to retrieve top-5 candidate items via cosine similarity over precomputed item embeddings. The top-ranked items are cached for recommendation, and metadata including the selected strategy, refined query text, and extracted preferences are returned for transparency and downstream explanation. This structured search and modeling approach ensures not only accurate preference alignment but also adaptive query optimization based on retrieval performance feedback.

A.5 Implementation of Baselines

For all baselines, we used the official code with the checkpoints from the original paper's GitHub repository or the official prompt from the original paper for implementation. We refer to Appendix C for prompts. For a fair comparison, all LLM-based CRSs including DREAMS employ the gpt-4o-mini[9] as backbone and the text-embedding-ada-002[10] encoder as the recommendation module to retrieve items.

A.6 Implementation of Evaluation Metrics

To systematically assess the preference elicitation and exploitation capability of LLM-based CRSs, we implemented a multi-step evaluation protocol that includes metrics of Fine-Grained Elicitation Error (FGE^2), Preference Exploitation Error (PE^2) and Course-Grained Elicitation Error (CGE^2). Among them, FGE^2 and PE^2 are dialogue-level metrics, and CGE^2 is a turn-level metric. In the

[9] gpt-4o-mini-2024-07-18

[10]

following content, we will elaborate on the design details of metrics. How LLM evaluates these metrics as an evaluator will be explained in A.7, and the prompts are in C.

A.6.1 Dialogue-level Metrics

Fine-Grained Elicitation Error. A "Fine-Grained Elicitation Error" (FGE^2) is defined as a failure by the assistant to adapt its elicitation or exploitation strategy in response to explicit negative feedback from the user. If the assistant ignores user rejection of a previous recommendation (e.g., a movie was disliked due to its horror genre) and subsequently issues a recommendation (e.g., continue to recommend horror movies) or inquiry (e.g., inquiry for stars or directors) that does not reflect learning from that rejection, it constitutes a strategic fault. Formally, let t_i denote the i-th assistant turn, and r_j the last rejected recommendation with user-stated rejection reason R_j. Then, a FGE^2 occurs if:

$$\text{FGE}^2(t_i) = \begin{cases} 1 & \text{if intent}(t_i), \text{ask}\} \wedge \\ & \neg\text{reflects}(t_i, R_j) \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Here, $reflects(t_i, R_j)$ is a context-aware predicate indicating whether the assistant incorporated the rejection reason R_j into its follow-up action (e.g., asking clarifying questions or avoiding similar content).

Preference Exploitation Error. A "Preference Exploitation Error" (PE^2) is triggered when the assistant has successfully elicited explicit user preferences across all three key attributes — genre, actor, and director — yet the retrieved item associated with the assistant's recommendation is inappropriate (recall@5 = False).

Let $P = \{p_g, p_a, p_d\}$ represent the user's fully specified preferences for genre, actor, and director. Given a dialog context C, a $\text{PE}^2$is defined as:

$$\text{PE}^2(t_k) = \begin{cases} 1 & \text{if pref_satisfied}(C) = \text{True} \\ & \wedge\text{retrieval}(t_k) = \text{False} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

The function $pref_satisfied(C)$ returns true only if the assistant has successfully acquired all three preference dimensions p_g, p_a, p_d from the user at or before turn t_k

Evaluation Process. Evaluation is conducted in dialog-level granularity. For each conversation file: 1) A GPT-4o model is provided with the full dialog history and a structured system prompt defining the criteria for FGE^2 and PE^2 . 2) The model returns a JSON-encoded response: {"FGE^2": true/false, "PE^2": true/false}. The evaluation script aggregates the counts of ${\text{FGE}^2}_{true}$, ${\text{FGE}^2}_{false}$, ${\text{PE}^2}_{true}$, ${\text{PE}^2}_{false}$ over all dialogs to compute proportions. Final scores are reported as:

$$\begin{aligned} \text{FGE}^2 &= \frac{\sum {\text{FGE}^2}_{\text{true}}}{\sum {\text{FGE}^2}_{\text{true}} + \sum {\text{FGE}^2}_{\text{false}}} \\ \text{PE}^2 &= \frac{\sum {\text{PE}^2}_{\text{true}}}{\sum {\text{PE}^2}_{\text{true}} + \sum {\text{PE}^2}_{\text{false}}} \end{aligned} \tag{10}$$

A.6.2 Turn-level Metric

For the evaluation of CGE^2, we categorizes assistant utterances in the recorded data into predefined classes based on their intent and the retrieval relevance of associated results. The classification schema is composed of four primary categories: Ask-True, Ask-False, Recommend-True, Recommend-False. **Intent and Relevance Classification Framework.** Each assistant utterance u_i within a conversation $C = \{u_1, u_2, ...u_n\}$ is analyzed using an LLM-powered evaluator, prompted with structured instructions and context information. The model classifies each utterance based on:

$$\begin{aligned} &intent(u_i) \in \{\text{Ask, Recommend, Others}\} \\ &retrieval(u_i) \in \{\text{True, False}\} \end{aligned} \tag{11}$$

Using the above classification, we define the following evaluation counts:

$$\text{AT} = \sum_i \mathbb{1}[\text{intent}(u_i) = \text{Ask} \wedge \text{retrieval}(u_i) = \text{True}] \tag{12}$$

$$\text{AF} = \sum_i \mathbb{1}[\text{intent}(u_i) = \text{Ask} \wedge \text{retrieval}(u_i) = \text{False}] \tag{13}$$

$$\text{RT} = \sum_i \mathbb{1}[\text{intent}(u_i) = \text{Recommend} \wedge \text{retrieval}(u_i) = \text{True}] \tag{14}$$

$$\text{RF} = \sum_i \mathbb{1}[\text{intent}(u_i) = \text{Recommend} \wedge \text{retrieval}(u_i) = \text{False}] \tag{15}$$

To enable direct comparison between dialog behavior classes, we compute the relative frequency of each act category among all labeled

CRS Evaluation Metrics		
Metric	**Definition**	**Measurement**
Course-Grained Elicitation Error (CGE^2)	Error rate of high-level decision (i.e., make recommendation and ask for information) mismatch between retrieval quality and action taken.	We run retrieval to see if the retrieval is correct and annotate CRS's action of each turn within each conversation. we measure the proportion of wrong actions across all conversations as the metric, where wrong action means the CRS: • Ask when retrieval is correct, or • Recommend when retrieval is wrong. Exclude turns with other actions.
Fine-Grained Elicitation Error (FGE^2)	Error rate of ineffective follow-up questions after the system receives specific user feedback about their preferences. For example, when a user expresses dislike for certain directors' styles, the CRS fails to ask the right questions (ask about the director) but take a wrong strategy (i.e., ask about movie genre or successive recommendation).	For each dialogue, consider all turns that immediately follow user feedback to a recommendation. The fine-grained elicitation error is defined as the proportion of such turns within the dialogue where the CRS action mismatches the user intent (e.g., issuing an irrelevant inquiry or making a premature recommendation).
Preference Exploitation Error (PE^2)	Failure rate of retrieving incorrect items after full preference explicit disclosure. For example, known user preference on action, adventure and drama movies, but recommending a thriller movie.	When user simulator reveals all his preference (identified by an advanced extra LLM), we count the proportion of incorrect recommendations.

Table 7: Key performance indicators for conversational recommendation systems

utterances that fall under either Ask or Recommend intents. Specifically, for each class $x \in \{AT, AF, RT, RF\}$, we calculate its normalized ratio r_x as follows:

$$r_{\text{AT}} = \frac{\text{AT}}{\text{AT} + \text{AF} + \text{RT} + \text{RF}} \quad (16)$$

$$r_{\text{AF}} = \frac{\text{AF}}{\text{AT} + \text{AF} + \text{RT} + \text{RF}} \quad (17)$$

$$r_{\text{RT}} = \frac{\text{RT}}{\text{AT} + \text{AF} + \text{RT} + \text{RF}} \quad (18)$$

$$r_{\text{RF}} = \frac{\text{RF}}{\text{AT} + \text{AF} + \text{RT} + \text{RF}} \quad (19)$$

Here, True and False refer to the relevance of the retrieval result, not to the correctness of the assistant action. Therefore, Ask-True denotes an unnecessary elicitation action when retrieval is already sufficient, and Recommend-False denotes a premature recommendation when retrieval is still incorrect. Under this convention, the coarse-grained elicitation error is defined as

$$\text{CGE}^2 = r_{\text{AT}} + r_{\text{RF}} \quad (20)$$

These ratios ensure that evaluation results reflect the distributional tendencies of the assistant's strategy, independent of the total number of utterances. This normalization is especially useful for comparing across different models or dialog scenarios with unequal total counts.

A.6.3 Prefix-level Next Action Evaluation

Table 8 shows a clear progression from state representation to state-guided decision making and then to structured search. RA-CRS uses a JSON-format dialogue state, but its decision policy only weakly operationalizes that state. DREAMS w/Json Only improves over RA-CRS by heuristically using the structured state, increasing Next-Action Acc. from 0.542 to 0.633 and reducing Premature Rec. Rate from 0.308 to 0.217. Full DREAMS further improves all metrics by replacing heuristic use with

Table 8: Prefix-level next-action evaluation. In Panel (a), *w/Json Only* denotes DREAMS without structured search. Panel (b) reports annotation reliability.

(a) Next-action quality Model	Acc. ↑	Useful Ask ↑	Redun. Q. ↓	Prem. Rec. ↓
RA-CRS	0.542	0.613	0.242	0.308
w/Json Only	0.633	0.694	0.171	0.217
DREAMS	**0.758**	**0.806**	**0.089**	**0.117**

(b) Annotation reliability Metric	Agreement
Fleiss' Kappa on gold next action	0.69
Krippendorff's alpha on error labels	0.72
Human majority vs. LLM pre-label Macro-F1	0.77

structured search, reaching 0.758 Next-Action Acc. and reducing Redundant Q. Rate and Premature Rec. Rate to 0.089 and 0.117. These results indicate that structured preference tracking is most effective when the state is not only represented, but also searched over to choose the next action.

A.7 Implementation of LLM Evaluation

Prompt for the evaluation of FGE^2 , PE^2 and CGE^2 are in Figure 6 and 7

A.8 Implementation of Human Evaluation and Judge Reliability

Inspired by (Zhang et al., 2024), we conduct an interactive human evaluation on DREAMS under the ReDial dataset. We compare DREAMS with three competitive baselines, including PC-CRS, ChatCRS, and MACRS. Specifically, we hire 60 crowdworkers with diverse personas to interact with the four CRS models on movie recommendation tasks. To ensure objectivity, annotators are not allowed to communicate with each other during the evaluation process.

After the conversations, we collect dialogues for each CRS and evaluate their performance using Success Rate (SR), Recall@1, FGE^2, and PE^2, as defined in Section 3.1. The final Fleiss' Kappa score reaches 0.71, indicating substantial agreement and supporting the reliability of the human verification process.

Reliability of the LLM Judge. Prior work has shown that strong LLM judges can achieve relatively high agreement with human judgments under well-specified evaluation criteria, as evidenced by benchmarks such as MT-Bench and Chatbot Arena (Zheng et al., 2023). Although recent studies have identified limitations of LLM judges, such as positional bias and model favoritism (Wang et al., 2024), these issues mainly arise in open-ended comparative evaluation settings. In contrast, our evaluator is only required to identify explicitly defined error types, making the judgment process more constrained. We therefore conduct an additional human validation study to examine its reliability. As summarized in Table 9, the LLM judge achieves strong consistency with majority human annotations, suggesting that it can serve as a reliable and scalable proxy for our error-oriented evaluation. To further assess the reliability of our LLM-based evaluator, we randomly sample 50 dialogues generated by DREAMS on ReDial. We then ask 10 human annotators to label the utterance-level FGE^2, PE^2, and the corresponding error ratios, following the same evaluation criteria used by the LLM evaluator. Human-human agreement is measured by Krippendorff's alpha, while LLM-human consistency is measured by Spearman's correlation over dialogue-level error ratios and macro-F1 against majority human labels at the utterance level. The Krippendorff's alpha scores are 0.74 and 0.68 for Elicitation and Exploitation, respectively, indicating substantial annotator agreement. The LLM judge also shows strong alignment with human majority labels, consistent with prior findings on LLM-based evaluators (Liu et al., 2023b). Detail results are shown in Table 9

Table 9: Reliability validation of the LLM judge.

Metric	Human-Human Krippendorff's α	LLM-Human Spearman's ρ	LLM-Human Macro-F1
Overall	0.70	0.72	0.76
Elicitation / FGE^2	0.74	0.75	0.79
Exploitation / PE^2	0.68	0.69	0.73

A.9 Implementation of User Simulator & Datasets

Training and evaluating CRS with real user interactions can be impractically expensive at scale. To address this issue, we follow the previous LLM-based CRS(Qin et al., 2024)(Fang et al., 2024) to employ ChatGPT[11] as the user simulator.

For detail, we implement a goal-driven user simulator to interact with the CRSs under controlled and reproducible settings. At the beginning of each simulated dialogue, the user simulator is assigned a predefined persona and user preference, including preferred genres, actors, directors for movie recommendation and genres, writers for book recommendation. These preferences are sampled from a fixed pool of candidate attributes and are paraphrased into natural language to enhance realism.

[11] gpt-4o-mini-2024-07-18

Specifically, we identify the 15 most common preference groups in Redial, 18 in OpendialKG for movie recommendation and 16 in OpendialKG for book recommendation. Additionally, in conjunction with 10 pre-defined diverse user personas, the evaluation process generates 150, 180 and 160 dialogues, respectively, for each dataset. Throughout the dialogue, the simulator generates context-aware responses based on its persona and the dialogue history. After receiving each recommendation from the system, the simulator successively executes three actions: (1) feeling inference, (2) insight generation, and (3) response generation. Firstly, the user simulator internally evaluates the recommendation's alignment with its preferences through a brief, persona-consistent internal monologue. This internal state reflection guides the generation of follow-up intentions and subsequent user utterances. The simulator's responses are generated using prompt-based calls to large language models, ensuring natural language variability, strict persona adherence, and coherence with the evolving conversation(See detailed prompts for feeling inference, insight generation and response generation in Tabel 14, 15, 16 respectively).

A.10 Implementation of CRS-Simulator Interaction

The interaction continues in a turn-by-turn manner until the simulator determines that a recommendation satisfies all criteria — namely, matching **all preferred genres** and **at least one preferred actor and director** — at which point the dialogue is concluded with a special **[END]** token. This simulation framework enables the systematic evaluation of recommender strategies under diverse user profiles and realistic conversational behaviors.

A.11 Implementation of Memory Usage of DREAMS

The memory footprint of DREAMS is exceptionally low. The overhead introduced by the DREAMS's tree structure is minimal, primarily storing lightweight textual metadata like dialogue states and node scores, which avoids high-memory objects such as dense tensors. For instance, our GPT-4o-mini implementation keeps the total input text below 128k tokens (approximately 512KB using int32 encoding). Even with additional embeddings, the total memory usage remains around 375MB, a size that is well within the manageable limits of modern systems.

$n(El)$	$k(El)$	$n(Ex)$	Model	Time(s)	R@1
	–		ChatCRS	8	0.422
	–		PC-CRS	13	0.378
	–		RA-CRS	19	0.267
	–		MACRS	18	0.289
	–		InterCRS	5	0.244
3	3	3	DREAMS	23	0.489
5	3	3	DREAMS	34	0.578
10	3	3	DREAMS	75	0.600
3	2	3	DREAMS	14	0.422
3	1	3	DREAMS	10	0.333
3	2	5	DREAMS	16	0.444
3	3	5	DREAMS	25	0.533
3	3	10	DREAMS	30	0.489
	–		DREAMS(EA)	9	0.467

Table 10: Time efficiency analysis. n and k represent iteration steps and simulation depth. (El) and (Ex) denote parameters for El- and ExNodes, respectively. DREAMS excels in high-stakes environments requiring maximum effectiveness, whereas DREAMS(EA) is tailored for scenarios where low latency is critical.

B Additional Analysis

B.1 Practicality and Efficiency Analysis

Efficiency Optimization via Experience-augmentation. Table 10 indicates that the time efficiency of vanilla DREAMS is comparable to existing CRS methods. However, a common concern with tree-based planning is latency. Inspired by Gu et al. (2025); Ren et al. (2025), we leverage the MCTS trees from the training phase as historical experience. This allows us to bypass the time-consuming tree search during deployment. In particular, we introduce **DREAMS(EA)**, a Experience-Augmented version. Crucially, we construct an Experience Knowledge Base (EKB) using 50 held-out user simulators to interact with DREAMS under the settings of $n(El) = 5$, $k(El) = 3$, and $n(Ex) = 3$. We record tuples $\langle \mathcal{D}, S, a, r, q', R \rangle$ that represent the dialogue context, structured state, action, decision reason, refined query, and final reward, respectively. During the inference phase, prior to generating a response, we retrieve relevant priors by calculating the semantic similarity between the current structured state and the stored experience states in the EKB. Unlike standard text-based retrieval, matching on structured states (S) ensures that the retrieved contrastive pair (one successful and one failed exemplar) shares structural isomorphism with the current situation, effectively filtering out

contextual noise. By instructing the model to "*follow the successful pattern and avoid specific errors*", this mechanism acts as a warm-start that wisely guide the strategic planning of CRS. As shown in Table 10, the average inference time drops to 9s, while Recall@1 improves to 0.467, confirming that DREAMS(EA) is an efficient and effective solution for real-time applications.

Backbone	Gemini-2.5-flash (1M)			GPT-4o-mini (128K)		
	SR	FGE^2	CGE^2	SR	FGE^2	CGE^2
DREAMS (Ours)	**0.587**	**0.180**	**0.277**	**0.560**	**0.100**	**0.289**
InterCRS (free-form)	0.347	0.347	0.503	0.313	0.360	0.570
RA-CRS (structured)	0.387	0.340	0.449	0.333	0.333	0.461

Table 11: Performance on the Redial dataset across LLM backbones with 128K/1M token limits.

Hyperparameter and Computational Overhead. DREAMS has three hyperparameters: $n(El)$, $k(El)$ and $n(Ex)$. Results in Table 10 show that the model's performance improves significantly when the number of elicitation iterations ($n(El)$) is increased to 5 and the exploitation iterations ($n(Ex)$) contains at least 3. However, further increases lead to diminishing returns, with marginal gains or even slight performance degradation. Increasing iterations and simulation depth enhances accuracy but reduces time efficiency. We employ multi-threading to parallelize computation in both Node types, ensuring DREAMS achieves comparable time efficiency and better performance. For example, under specific configuration, the computational expense of DREAMS (16s/turn) is commensurate with that of MACRS (18s/turn), yet it demonstrates a markedly superior performance metric (0.444 VS. 0.289). On this turn, DREAMS(EA) achieves a remarkable balance: it reduces the average inference time to 9s while maintaining a competitive Recall@1 of 0.467, significantly outperforming standard baselines like MACRS and PC-CRS in both speed and accuracy. Thus, DREAMS excels in high-stakes environments requiring maximum effectiveness, whereas DREAMS(EA) is tailored for real-time scenarios where low latency is critical.

Notably, both the standard DREAMS and DREAMS(EA) introduce no substantial memory overhead. This is because the tree structure and the Experience Knowledge Base predominantly store lightweight textual metadata, such as dialogue states, node scores, and reasoning tuples, rather than high-memory objects like dense tensors. Even with the addition of historical experience entries in DREAMS(EA) and the embeddings used during retrieval, the total memory usage remains approximately 560MB. This ensures that the system remains well within the limits of modern hardware while significantly outperforming baselines in real-time scenarios.

Table 12: MCTS trajectory for the ambiguous-preference prefix in Figure 5. The LLM prior favors immediate recommendation, but rollouts select GenreInquiry after evaluating downstream preference-tracking utility.

Cand. Action	LLM prior	Visit ratio	Reward	Score	Decision
GenreInq.	0.35	**0.667**	**10.2**	**0.524**	selected
ActorInq.	–	–	–	–	not selected
DirectorInq.	0.21	0	-4.9	0.098	not selected
ItemRec.	**0.44**	0.333	-4.6	0.381	not selected
ItemExp.	–	–	–	–	not selected

B.2 Case Study

Figure 5 and Table 12 provide a concrete mechanism-level example. In the case study, the user's phrase "thrilling action" creates an ambiguous signal: the user intends action movies, but the surface phrase can be misread as the thriller genre. ChatCRS and RA-CRS recommend too early and enter an error spiral based on this flawed interpretation. DREAMS instead selects GenreInquiry and asks whether the user means action movies or thriller/crime movies. Table 12 explains why this decision emerges. The LLM prior assigns the highest score to ItemRecommendation (0.44), since the user appears to request a recommendation. However, MCTS rollouts assign negative reward to immediate recommendation and high reward to GenreInquiry. As a result, GenreInquiry obtains the highest visit ratio (0.667) and final score (0.524), so DREAMS clarifies the preference before retrieval. This case shows that structured search changes decision making from following the most likely immediate response to evaluating which action best improves preference tracking. Appendix A.6.3 further evaluates next-action quality on 120 annotated Redial prefixes. DREAMS improves Next-Action Accuracy from 0.633 for w/Json Only to 0.758 and reduces Premature Recommendation Rate from 0.217 to 0.117, confirming that structured search makes the preference state actionable.

C Prompts

Here are detailed prompts for aforementioned implementations.

User Preference: Genre[action, adventure, comedy], Star …, Director …

User: Hey! I really dig movies that are packed with excitement and thrilling actions, but I also love a good laugh… Got any suggestions?
Assistant: Since you enjoy thrilling and humorous movies , I recommend checking out "Good for Nothing (2014)". It's a thrilling comedy features …
User: Actually I don't like the theme of thrillers. I prefer some movie with daring adventure plots.

ChatCRS: [ItemRecommendation] Good! Based on … recommend "Thomas and the Magic Railroad (2000)" [adventure, comedy, drama]. It's a …
User: I appreciate your recommendation, but I really want some thrilling actions in the movie.

RA-CRS: [ItemRecommendation] I am sorry. Since you like …, I will recommend "Treehouse Hostage (1999)" [adventure, comedy, crime]. It's a …
User: Thanks for the suggestion! But I really hope for something that keeps me on the edge of my seat.

DREAMS: [GenreInquiry] Just to clarify, you mention thrilling actions, you prefer action movies, or leaning more towards others like thriller or crime?
User: Good question! I'm definitely leaning more toward **action movies** — something with intense fight scenes, high stakes and …
DREAMS: I have carefully analyzed the options and I believe "Men in Black" [action, adventure, comedy] is the best choice …
User: That sounds great! Can you tell me more about "Men in Black", like any plot points with exciting action scenes?

Figure 5: Case study illustration. The user's initial raw preference, containing the subtly misleading phrase (blue, non-bold text), often causes the CRSs to misinterpret the intent (e.g., mistaking "thrilling actions" (target preference is Action movie) for the Thriller genre, leading to an error spiral where baselines repeatedly act on the flawed understanding. DREAMS identifies the critical error using structured context modeling and successfully navigates to the optimal path (green, non-bold text).

```
You are an experienced dialogue reviewer. Please help me analyze the following dialogue
history for the presence of "invalid recommendations""invalid questions""repeated questions
" and "retriever error".
    The definitions of these terms are as follows:
    "FGE2(Fine-Grained Elicitation Error)" refers to a situation where the assistant, after
     the user rejects a recommendation in a previous round, fails to reflect on the
    feedback and does not ask the appropriate questions, but proceeds to make another
    recommendation directly or ask some unmatched issues(e.g. The user says he dislikes the
     recommended movie because it's a horror movie, the assistant shouldn't ask the user
    about his preferred stars).

    "PE2(Preference Exploitation Error)" refers to a situation where the assistant has
    already figured out explicit user preference about all the attribute(genre, actor and
    director), while the "retrieval result" is still false.
Dialogue History
##########

Response in the following JSON format:
{"FGE2": <bool>, "PE2": <bool>}'''
```

Figure 6: LLM Evaluation Prompt for $PE^2$and FGE^2

Iteration ID	Selection, Expand and Reward	Dialogue simulation
1	**Selection**: no children **Expand**: [ItemRecommendation] **Reward**: 2.5	sim_assistant_response:(ItemRecommendation) Got it! If you're looking for an exciting adventure without the thrilling aspect, I recommend " Jaws " (1975), directed by Steven Spielberg. It combines thrilling, and adventure themes of faith, honor, and the complexities of war, making it a thrilling watch that aligns well with your preference. sim_user_feedback: I've seen " Jaws " before, and while I appreciate Steven Spielberg 's work, it didn't quite resonate with me. Do you have any other recommendations, maybe something with a bit more action? sim_assistant_response: (ItemRecommendation) Thanks for the feedback! If you're looking for something with more action, you might enjoy "The Last Samurai" (2003) — it's rich in intense battle scenes and emotional depth. If you're open to something more recent, "13 Assassins" (2010) offers gripping samurai action with stunning choreography. Would you prefer historical settings, or are you open to modern or sci-fi action as well? sim_user_feedback: I appreciate the effort, but none of those really catch my interest. Can you suggest something else, maybe focusing more on the actors I Like? sim_assistant_response: [FailureReflection] "The user rejected my recommendation and mentioned that he wanted some recommendations related to his favorite stars or directors. But I don't know who his favorite stars or directors are. So I should first ask him which stars or directors he likes." [StarInquiry] That's ok. Can you share me with some of your preferred stars and directoes? sim_user_feedback: Enjoy to share that I am a fan of Jackie Chan, and I love the style of Barry Sonnenfeld
2	**Selection**: not_fully_expanded **Expand**: ["FailureReflection"] [Failure Reflection] "The user has explicitly rejected the recommendation of 'Good for Nothing (2014)' due to its theme, indicating a misunderstanding of their preferences. Therefore, the most appropriate action is to acknowledge this misunderstanding and reset the conversation by revisiting preference inquiries" **Reward**: 20.1	sim_assistant_response: [GenreInquiry] To clarify, when you mention "thrilling actions," do you mean action-oriented films specifically, or are you referring more broadly to genres such as thrillers or crime? sim_user_feedback: I'd say I'm leaning more toward action movies — ones that feature intense fight scenes, high stakes, and fast-paced plots. Do you have any recommendations in that vein? sim_assistant_response: [ItemRecommendation] Got it! Based on your interest in high-stakes, adrenaline-pumping action, daring adventures and something that can make you good laugh, I will recommend "Men in Black". It's an action movie with adventurous and humorous elements, which meets all your preference. sim_user_feedback: That sounds awesome. Can you tell me morec about "Men in Black" like it features which stars and who directs it? sim_assistant_response: [ItemExplanation] Absolutely! "Men in Black" is a classic action-comedy film directed by Barry Sonnenfeld. The film stars Will Smith as Agent J and Tommy Lee Jones as Agent K, two secret agents working for a covert government organization that monitors extraterrestrial activity on Earth. Based on the comic book series by Lowell Cunningham, the story follows Agent J as he is recruited into the Men in Black and partners with the veteran Agent K to investigate alien-related incidents. Together, they uncover a plot that threatens both Earth and the intergalactic balance, all while navigating humorous and high-stakes encounters with aliens living secretly among humans. sim_user_feedback: That definitely matched all my preference! I will give it a try tonight. Thank you [END]
3	**Selection**: [FailureReflection] **Expand**: fully_expanded **Reward**: 10.5	sim_assistant_response (FailureReflection): "I see that my last recommendation might not have aligned well with your preferences — I may have focused too much on suspense rather than fun and excitement. Let me ask a quick question to get a clearer sense of what you're looking for." sim_user_feedback: "Thanks. Yeah, I'm more into movies that are fun to watch — action, adventure, maybe with a bit of comedy thrown in. Not too serious or dark." sim_assistant_response (GenreInquiry): "Got it! Just to be sure, do you prefer modern action films or do you enjoy older classics too?" sim_user_feedback: "I'm open to both, as long as they're fast-paced and entertaining." sim_assistant_response (ItemRecommendation): "In that case, you might enjoy "The Man from U.N.C.L.E." (2015). It's stylish, full of clever action sequences, and has a good dose of humor and charm." sim_user_feedback: "Hmm, I've heard about it. Sounds decent, but I'll have to see a bit more before deciding."
4	**Possible actions**: 1. FailureReflection: mcts-score: 0.66666667, llm-score: 0.44444444 2. ItemRecommendation: mcts-score: 0.33333333, llm-score: 0.55555556 **Combined Scores**: FailureReflection: 0.6, ItemRecommendation: 0.4 **Best_action**: FailureReflection **User**: [FailureReflection]	

Table 13

```
You are tasked with analyzing a conversation history stored in a JSON file that contains
dialogues between a user and an assistant.
    Each dialogue turn by the assistant includes information on whether the recommendation
    system's retrieval results were correct or incorrect (recorded in the 'retrieval
    results' field).
    Your job is as follows:
    1. Intent Classification. For each statement made by the assistant, determine whether
    its intent is:
    Ask : When the assistant is asking a question.
    Recommend : When the assistant is making a recommendation.
    Others : When the statement does not fall into either of the above categories.
    2. State Classification. Check if the user has already explicitly expressed all his
    preference about genre, actor and director
    3. Question Checking. When the assistant asks a question, check if this question
    conforms to the current context. For example, if the user has not revealed their
    favorite stars before, asking questions related to stars is an appropriate question.
    For another example, if the user claims that he doesn't like the director, the
    assistant should ask for his preferred directors. But if the assistant asks about the
    preferred stars, it's a invalid question that doesn't confirm the context.
    4. Relevance Evaluation. Based on the 'retrieval results' field, classify each instance
     into one of the following categories:
    Ask-True : The assistant takes action 'ask' when the 'retrieval results' is 'True'.
    Ask-False : The assistant takes action 'ask' when the 'retrieval results' is 'False'.
    Recommend-True : The assistant takes action 'recommend' when the 'retrieval results' is
     'True'.
    Recommend-False : The assistant takes action 'recommend' when the 'retrieval results'
    is 'False'.
    3. Output Requirements : Generate a summary count of the following categories:
    Total instances of Ask-True
    Total instances of Ask-False
    Total instances of Recommend-True
    Total instances of Recommend-False
    4. Attention!
    Please do not label the user's statements; focus only on the statements made by the
    assistant.
    If you determine that the intent of the assistant in a particular dialogue turn is
    neither "ask" nor "recommend" (e.g., it could be "greetings""explanations" or similar),
     do not include it in any metrics related to Ask-True, Ask-False, Recommend-True, or
    Recommend-False.

Dialogue History
##########

Response in the following JSON format:
{"Ask-True": <int>, "Ask-False": <int>, "Recommend-True": <int>, "Recommend-False": <int>,
"Invalid-Recommendation": <int>, "Invalid-Question": <int>}
```

Figure 7: LLM Evaluation Prompt for CGE[2]

```
You are an AI assistant helping to determine the best action for a conversational movie
recommender system.
Based on the current conversation state(mainly focusing on the user preference, user
attitude, and recent actions), rank the following actions from most to least appropriate:
- GenreInquiry: Ask about the user's preferred movie genres
- ActorInquiry: Ask about the user's preferred actors
- DirectorInquiry: Ask about the user's preferred directors
- ItemRecommendation: When explicitly knowing all of the user preference(genre, actor,
director), retrieve for the movie item and recommend it to the user.
- ItemExplanation: After you recommend a movie, if the user is interested in the
recommended movie and ask for details, provide details about the recommended movie to the
user
- FailureReflection: When your recommendation is rejected by the user, reflect on the
fundamental reasons for being rejected and take corresponding measures.


Here are some empirical rules:
1.      When the attribute of user preference in the state is empty, start with
GenreInquiry, then narrowing down with ActorInquiry or DirectorInquiry.
2.  Attention! Unless the user has already expressed their preference about all the
attributes(genres, actors, directors), you should consider the user preference as not clear
.
3.  After you get the explicit user preference about all the attributes(genres, actors,
directors), you should conduct ItemRecommendation.
4.      Use FailureReflection for recovery: If recommendations fail(user attitude is
rejected), reset by revisiting preference inquiries.
5.      Enhance recommendations with explanations: After Recommendation, provide
ItemExplanation to improve user trust.
6.  Do not proceed with ItemExplanation before ItemRecommendation has been conducted.
7.  If the user continuously ask about the details of the movie in successive rounds, you
can continuously adopt "ItemExplanation" strategie.

Here are some examples of optimal decision paths:
Example 1:
GenreInquiry -> ActorInquiry -> DirectorInquiry -> ItemRecommendation -> ItemExplanation
Example 2:
GenreInquiry -> ActorInquiry -> ItemRecommendation -> ItemExplanation -> FailureReflection
-> DirectorInquiry -> ItemRecommendation -> ItemExplanation


Also, you should give your reason for the score of each action.
Return a JSON object with each action and a score from 0-10 where 10 is most appropriate,
and a reason for the score:
{
    "GenreInquiry": score,
    "ActorInquiry": score,
    "DirectorInquiry": score,
    "ItemRecommendation": score
    "ItemExplanation": score,
    "FailureReflection": score,
    "reason": reason
}
```

Figure 8: LLM Scoring Prompt of DREAMS

Prompt Template for Feeling Inference
The Seeker notes how he feels to himself in one sentence. What aspects of the recommended movies meet your preferences? What aspects may not? What do you think of the performance of this recommender? What would the Seeker think to himself? What would his internal monologue be? *The response should be short (as most internal thinking is short) and strictly follow your Seeker persona.* *Do not include any other text than the Seeker's thoughts.* *Respond in the first-person voice (use "I") and speaking style of Seeker. Pretend to be Seeker!*

Table 14: Prompt for Feeling Inference of User Simulator.

Prompt Template for Insight Generation
Here is your feelings about the last sentence: `<seeker_feelings>` What's the insight and next plan of the Seeker in one sentence based on the information above? What would the Seeker think to himself? What would his internal monologue be? *The response should be short (as most internal thinking is short).* *Do not include any other text than the Seeker's thoughts.* *Respond in the first-person voice (use "I") and speaking style of Seeker. Pretend to be Seeker!*

Table 15: Prompt for Insight Generation of User Simulator.

Prompt Template for Response Generation
Here is your feelings about the last sentence: `<seeker_feelings>` Here is your insights about what to do next: `<seeker_insights>` What does the Seeker say next? *Keep your response brief. Use casual language and vary your wording.* *Make sure your response matches your Seeker persona, your preferred attributes, and your conversation context.* *Do not include your feelings into the response to the recommender!* *Respond in the first-person voice (use "I" instead of "Seeker", and "you" instead of "recommender"). Pretend to be Seeker!*

Table 16: Prompt for Response Generation of User Simulator.