SAPIENT: Mastering Multi-turn Conversational Recommendation with Strategic Planning and Monte Carlo Tree Search

Hanwen Du[♣] **Bo Peng**[♣] **Xia Ning**[♣♡◇✉]
[♣]Department of Computer Science and Engineering, The Ohio State University, USA
[♡]Department of Biomedical Informatics, The Ohio State University, USA
[◇]Translational Data Analytics Institute, The Ohio State University, USA
{du.1128,peng.707,ning.104}@osu.edu

Abstract

Conversational Recommender Systems (CRS) proactively engage users in interactive dialogues to elicit user preferences and provide personalized recommendations. Existing methods train Reinforcement Learning (RL)-based agent with greedy action selection or sampling strategy, and may suffer from suboptimal conversational planning. To address this, we present a novel Monte Carlo Tree Search (MCTS)-based CRS framework SAPIENT. SAPIENT consists of a conversational agent (S-agent) and a conversational planner (S-planner). S-planner builds a conversational search tree with MCTS based on the initial actions proposed by S-agent to find conversation plans. The best conversation plans from S-planner are used to guide the training of S-agent, creating a self-training loop where S-agent can iteratively improve its capability for conversational planning. Furthermore, we propose an efficient variant SAPIENT-e for trade-off between training efficiency and performance. Extensive experiments on four benchmark datasets validate the effectiveness of our approach, showing that SAPIENT outperforms the state-of-the-art baselines. Our code and data are accessible through https://anonymous.4open.science/r/SAPIENT/.

1 Introduction

Conversational Recommender Systems (CRSs) are developed to proactively engage users with interactive dialogues to understand user preferences and provide highly personalized recommendations [1, 2]. For example, on an online dining platform such as Yelp [2], CRS can chat with users through natural language dialogues (e.g., ask a question like, "What is your preferred food type?") and recommend products that best match the users' preferences expressed in the conversation. Among different settings of CRS [3–5], the Multi-turn Conversational Recommendation (MCR) setting [2, 6, 4] is popular as it can interact/communicate with users multiple times (i.e., multiple turns) to iteratively learn user preferences [7, 8].

A key of MCR is to decide what action (asking a question on specific attribute values or recommending specific items) to take at each conversational turn—a conversational turn consists of the CRS taking an action and the user responding to that action—to effectively elicit information on user preferences and make personalized recommendations [7, 9]. To achieve this, previous methods formulated MCR as a Marchovian Decision Process (MDP) [10], and trained policy-based [3, 2] or value-based [4, 11] agents via Reinforcement Learning (RL) to learn conversation strategies. Despite promising, these methods could suffer from myopic actions and limited planning capability due to the following reasons. First, they base their planning solely on observations of the current state (e.g., items that the user indicates a negative preference for) without exploring potential future states. As a result, they

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

could take myopic actions [12, 13]. Second, they generate conversation trajectories, also referred to as conversation plans, by sequentially sampling actions, and thus could suffer from the cumulative error, especially when generating long trajectories for planning [14, 15].

To address these limitations, we present a novel MCTS-based MCR framework—Strategic Action Planning with Intelligent Exploration Non-myopic Tactics, referred to as `SAPIENT`. `SAPIENT` comprises a conversational agent, referred to as `S`-agent, where `S`-agent utilizes an MCTS-based algorithm, referred to as `S`-planner, to plan conversations. `S`-agent builds a global information graph and two personalized graphs with dedicated graph encoders to extract the representation of the conversational states, and synergizes a policy network and a Q-network to decide specific actions based on the learned state representations. `S`-planner leverages MCTS [16, 17] to simulate future conversations with lookahead explorations. This non-myopic conversational planning process ensures `S`-planner can strategically plan conversations that maximize the cumulative reward, instead of greedily selecting actions based on immediate reward. The best conversation plans with the highest cumulative rewards found by `S`-planner are used to guide the training of `S`-agent. In this way, `S`-agent can engage in a self-training loop [18]—collecting trajectories from multiple conversation simulations and training the `S`-agent on selected, high-rewarded trajectories—to iteratively improve its planning capability. During inference, `S`-agent can directly make well-informed decisions without `S`-planner, since it inherits the `S`-planner's expertise in strategic and non-myopic planning.

To make MCTS scalable w.r.t. the size of items and attributes, we introduce a hierarchical action selection process [19], and two action types: `ask` and `rec`. At each turn, instead of searching over all the items and attribute values, `S`-planner builds a conversational search tree (Figure 2) that only searches over the two action types and uses the Q-network to decide the specific action, thus greatly reducing the search space.

We evaluate `SAPIENT` against 9 state-of-the-art CRS baselines, and show `SAPIENT` significantly outperforms baselines on 4 benchmark datasets. Our case study also shows that the action strategies of `SAPIENT` are beneficial for information seeking and recommendation success in the conversations.

Furthermore, we develop an efficient variant of `SAPIENT`, denoted as `SAPIENT`-e. Different from `SAPIENT`, which is trained on selected, high-rewarded trajectories, `SAPIENT`-e consumes all trajectories found by `S`-planner for training via a listwise ranking loss. As a result, `SAPIENT`-e requires less cost of collecting training trajectories compared to `SAPIENT`, and enables superior efficiency.

2 Method

We introduce `SAPIENT`, a Monte Carlo Tree Search (MCTS)-based MCR framework that achieves strategic and non-myopic conversation planning. `SAPIENT` formulates MCR as an MDP with a hierarchical action selection process (Section 2.1). A conversational agent (`S`-agent) observes the current state and decides the actions in each conversational turn (Section 2.2), and a conversational planner (`S`-planner) performs non-myopic conversational planning with an MCTS-based algorithm (Section 2.3). `S`-agent is trained with guidance from `S`-planner to iteratively refine its capability for conversational planning (Section 2.4). An overview of the `SAPIENT` framework is in Figure 1 and the training algorithm is in Appendix H. We summarize all the notations in this paper in Appendix A.

2.1 MDP Formulation for MCR

We formulate MCR as an MDP where `S`-agent can be trained in an RL environment to learn to plan conversations strategically. For each user u, the MDP environment $\mathcal{M}(u)$ is defined as a quintuple $\mathcal{M}(u) = \{\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma\}_u$ (index u is dropped when no ambiguity arises), where $\mathcal{S}$ denotes the state space, which summarizes all the information about the conversation and the user; $\mathcal{A}$ denotes the action space, which includes asking (`ask`) for specific attribute types and their respective values, or recommending (`rec`) specific items; $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ denotes the transition to the next state after taking an action from the current state; $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ denotes the immediate reward function after taking an action at the current state; and $\gamma \in (0, 1)$ denotes the discount factor. For hierarchical action selection [19], `S`-agent first chooses an action type $o_t \in \{\texttt{ask}, \texttt{rec}\}$ at each conversation round, indexed by t, then chooses the objective of that action type.

State For the t-th turn, we define the state $s_t \in \mathcal{S}$ as a triplet $s_t = (\mathcal{P}_t^+, \mathcal{P}_t^-, \mathcal{V}_t^-)$, where $\mathcal{P}_t^+$ denotes all the attribute values that the user has accepted until the t-th turn, $\mathcal{P}_t^-$ and $\mathcal{V}_t^-$ denote all the attribute values and items that the user has rejected until the t-th turn. The state is initialized as s_0 when the user specifies preference on an attribute type $y_0 \in \mathcal{Y}$ and its corresponding attribute value $p_0 \in \mathcal{P}$, and transitions to the next states as the conversation continues.

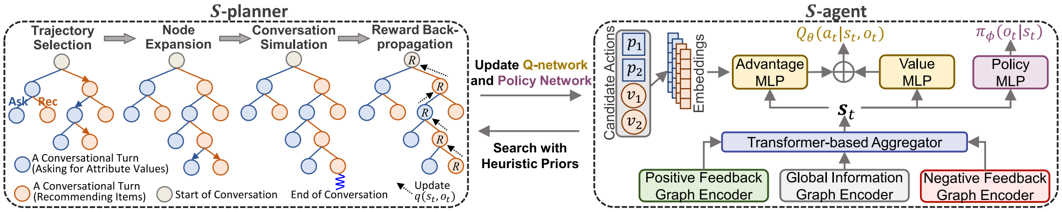

Figure 1: SAPIENT consists of a conversational agent (S-agent) an conversational planner (S-planner). S-planner leverages MCTS to perform non-myopic conversation planning based on the heuristics from S-agent. The best conversation plans found by S-planner are used to guide the training of S-agent, allowing S-agent to iteratively refine its capability for conversational planning.

Action The action a_t refers to asking a specific attribute value (`ask`) or recommending a specific item (`rec`) at the t-th turn. Here, we adopt a hierarchical action selection process: we first use a new policy network $\pi_\phi(o_t|s_t)$ to decide the action type $o_t \in \{\texttt{ask}, \texttt{rec}\}$ from the current state s_t, and then use a new Q-network $Q_\theta(a_t|s_t, o_t)$ to decide the specific action a_t according to the action type o_t. The action space (at the current state s_t) $\mathcal{A}_{s_t} = \{\mathcal{P}_t^c, \mathcal{V}_t^c\}$ contains all the candidate items and attribute values. The Q-network $Q_\theta(a_t|s_t, o_t)$ only selects an action from a sub action space $\mathcal{A}_{s_t,o_t}$: when $o_t = \texttt{ask}$, $\mathcal{A}_{s_t,o_t} = \mathcal{P}_t^c$; when $o_t = \texttt{rec}$, $\mathcal{A}_{s_t,o_t} = \mathcal{V}_t^c$.

Transition Transition occurs from the current state s_t to the next state s_{t+1} when the user responds to the action a_t (accepts or rejects items/attribute values). Candidate item set are narrowed down to the remaining items that still satisfy the user's preference requirement, and attribute values asked at turn t are excluded from the candidate attribute value set. More details are in Appendix D.

Reward We denote the immediate reward at the t-th conversation turn as r_t, and the cumulative reward for each conversation is calculated as $\sum_{t=1}^{T} \gamma^t r_t$. Intuitively, a positive reward is assigned when the user accepts the items or attribute values, and a negative reward is assigned when the user rejects the items or attribute values. Details on the reward function are available in Appendix E.

2.2 S-agent

S-agent comprises three components: the state encoder, the policy network, and the Q-network. The state encoder adopts graph neural networks to generate the state representation, as detailed in Appendix F. This state representation is then utilized by both the policy network and the Q-network to decide the action type and the specific actions in each conversational turn, as detailed in Appendix G.

2.3 S-planner

S-planner adopts an MCTS-based planning algorithm to simulate conversations and find the best conversation plan for each user, strategically balancing exploration and exploitation to efficiently expand a search tree [16, 17]. Specifically, each node in the tree represents a state s_t, the root node s_0 represents the initial state where the user specifies preference on an attribute type and its corresponding value, and the leaf node represents the end of the conversation (success or fail). Each edge between nodes s_t and s_{t+1} represents an action type $o_t \in \{\texttt{ask}, \texttt{rec}\}$ and the transition from the current state s_t to the next state s_{t+1} after choosing an action type o_t and a specific action a_t. For each action type o_t, S-planner maintains a function $q(s_t, o_t)$ of s_t and o_t as the expected future reward of selecting action type o_t at the state s_t. For each user u, S-planner simulates N different conversation plans (also referred to as trajectories), and the trajectory for the i-th simulation is denoted as $\tau_i^{(u)}$. The search tree is built in four stages, as summarized below and detailed in Appendix I:

- Trajectory selection: S-planner traverses from the root to leaves over the current tree to select the most promising trajectory that is likely to obtain a high cumulative reward.
- Node expansion: S-planner initializes two children nodes (`ask` and `rec`) to the leaf node on the selected trajectory to expand the tree.
- Conversation simulation: S-planner simulates future conversations between SAPIENT and the user, starting from the expanded node and foresees how the future conversation will unfold.
- Reward back-propagation: S-planner updates the expected future reward $q(s_t, o_t)$ of action type o_t along the trajectory using the cumulative reward of the current conversation.

2.4 Guiding S-agent with S-planner

To empower S-agent with advanced planning capability, we use the best conversation plan (the plan with the maximum cumulative reward) found by S-planner to guide the training of the policy network and the Q-network. This process creates a self-training loop [18] that enables S-agent to iteratively refine its planning capability. To avoid biased estimation from training on consecutive, temporally correlated actions [20], we store the experiences $e_t = (s_t, o_t, a_t, r_t, s_{t+1}, o_{t+1})$ at each turn t (s_{t+1} and o_{t+1} are required for the target Q-network to estimate the Q value from the next state) from the best plans to the memory $\mathcal{D}$, and use Prioritized Experience Replay (PER) [21] to sample a batch of experiences from the memory $\mathcal{D}$ to update the policy network and the Q-network. To improve efficiency, we further propose a variant `SAPIENT`-e. Instead of using only the highest-rewarded trajectories, `SAPIENT`-e makes full use of all the trajectories found by `S`-planner, and thus greatly reduces the cost of collecting enough trajectories. More details are available in Appendix J.

3 Experimental Settings

Datasets We evaluate `SAPIENT` on 4 benchmark datasets: Yelp [2], LastFM [2], Amazon-Book [22] and MovieLens [23]. Dataset details are in Appendix K.1.

User Simulator Training and evaluating CRS with real-world user interactions can be impractically expensive at scale. To address this, we adopt the user simulator approach [2] and simulate a conversation for each user as detailed in Appendix K.2. Note that this user simulator is widely adopted in the literature [2, 4, 11, 24, 25] and researches show the simulations are of high quality and suitable for evaluation purposes [9, 4, 11], allowing for large-scale evaluations at low cost.

Evaluation Metrics Following the literature [4, 11], the Success Rate (SR) is adopted to measure the ratio of successful recommendations within T_{max} turns; Average Turn (AT) to evaluate the average number of conversational turns; and hDCG [4] to evaluate the ranking order of the ground-truth item among the list of all the recommended items. Details for hDCG calculation are in Appendix K.3.

Baselines and Implementation Details We choose 9 state-of-the-art baselines for a comprehensive evaluation, including a Large Language Model (LLM) baseline CORE [26]. Baseline details are in Appendix K.4. Implementation details of `SAPIENT` are in Appendix K.5.

4 Experimental Results

4.1 Overall Performance Comparison

We compare `SAPIENT` with baselines and report the results in Table 3. Our main conclusions are **(1)**`SAPIENT` achieves consistent improvement over baselines in terms of all metrics on all the datasets, with an average improvement of 9.1% (SR), 6.0% (AT) and 11.1% (hDCG) compared with the best baseline. **(2)** `SAPIENT` substantially outperforms baselines in datasets demanding strong strategic planning capacity from the CRS. **(3)** `SAPIENT`-e outperforms all baselines on recommendation performance. More details are available in Appendix K.6.

4.2 Efficiency Analysis

We show that even with MCTS, the training efficiency of `SAPIENT` and `SAPIENT`-e is highly comparable to the baselines, as detailed in Appendix K.7.

4.3 Ablation Study

We conduct ablation studies to validate the effectiveness of the key components as detailed in Appendix K.9. Our main conclusions are: **(1)** Each component—$\mathcal{G}$, $\mathcal{G}^+$, $\mathcal{G}^-$—from the state encoder are crucial for state encoding. **(2)** Both the policy network and the Q-network are crucial for hierarchical action selection. **(3)** Removing `S`-planner and training the `S`-agent on sampled trajectories as in [4, 11] significantly degrades the performance.

4.4 Hyperparameter Sensitivity

To explore how `SAPIENT` balances exploration and exploitation in conversational planning, we evaluate the performances under different exploration factor w, and find that `SAPIENT` performs worse with little exploitation (small w), but too much exploration (large w) does not undermine the performances. To determine the appropriate rollout number N, we analyze the performances with different N, and find that the performance notably improves when increasing N from 1 to 20 and marginally improves after $N > 20$, This suggests that setting $N = 20$ can strike a good balance between efficiency (small N) and performance (large N). More details are in Appendix K.10.

4.5 In-depth Analysis

To gain insight into the conversation planning capacity of `SAPIENT`, we provide an analysis on the action strategies of `SAPIENT` (Appendix K.11) and a case study (Appendix K.12) to show `SAPIENT` can strategically take actions that are helpful for information seeking and recommendation success.

5 Conclusion

We present `SAPIENT`, a novel MCTS-based CRS framework with strategic and non-myopic planning tactics in conversational turns. `SAPIENT` adopts a hierarchical action selection process, builds a conversation search tree with MCTS, and uses the highest-rewarded conversation plan to train the `S`-agent. Extensive experiments on 4 benchmark datasets validate the effectiveness of `SAPIENT`.

References

[1] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. Towards conversational recommender systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 815–824, 2016. URL https://doi.org/10.1145/2939672.2939746.

[2] Wenqiang Lei, Xiangnan He, Yisong Miao, Qingyun Wu, Richang Hong, Min-Yen Kan, and Tat-Seng Chua. 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, 2020. URL https://doi.org/10.1145/3336191.3371769.

[3] Yueming Sun and Yi Zhang. Conversational recommender system. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 235–244, 2018. URL https://doi.org/10.1145/3209978.3210002.

[4] Yang Deng, Yaliang Li, Fei Sun, Bolin Ding, and Wai Lam. 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, 2021. URL https://doi.org/10.1145/3404835.3462913.

[5] Zhankui He, Zhouhang Xie, Rahul Jha, Harald Steck, Dawen Liang, Yesu Feng, Bodhisattwa Prasad Majumder, Nathan Kallus, and Julian Mcauley. Large language models as zero-shot conversational recommenders. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, page 720–730, 2023. URL https://doi.org/10.1145/3583780.3614949.

[6] Wenqiang Lei, Gangyi Zhang, Xiangnan He, Yisong Miao, Xiang Wang, Liang Chen, and Tat-Seng Chua. Interactive path reasoning on graph for conversational recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2073–2083, 2020. URL https://doi.org/10.1145/3394486.3403258.

[7] Zuohui Fu, Yikun Xian, Yongfeng Zhang, and Yi Zhang. Tutorial on conversational recommendation systems. In *Proceedings of the 14th ACM Conference on Recommender Systems*, pages 751–753, 2020. URL https://doi.org/10.1145/3383313.3411548.

[8] Dietmar Jannach, Ahtsham Manzoor, Wanling Cai, and Li Chen. A survey on conversational recommender systems. *ACM Computing Surveys (CSUR)*, 54(5):1–36, 2021. URL https://doi.org/10.1145/3453154.

[9] Wenqiang Lei, Xiangnan He, Maarten de Rijke, and Tat-Seng Chua. Conversational recommendation: Formulation, methods, and evaluation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2425–2428, 2020. URL https://doi.org/10.1145/3397271.3401419.

[10] Richard Bellman. A Markovian decision process. *Journal of Mathematics and Mechanics*, pages 679–684, 1957. URL http://www.jstor.org/stable/24900506.

[11] Yiming Zhang, Lingfei Wu, Qi Shen, Yitong Pang, Zhihua Wei, Fangli Xu, Bo Long, and Jian Pei. Multiple choice questions based multi-interest policy learning for conversational recommendation. In *Proceedings of the ACM Web Conference 2022*, pages 2153–2162, 2022. URL https://doi.org/10.1145/3485447.3512088.

[12] Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/d8e1344e27a5b08cdfd5d027d9b8d6de-Paper.pdf.

[13] Deborah Cohen, Moonkyung Ryu, Yinlam Chow, Orgad Keller, Ido Greenberg, Avinatan Hassidim, Michael Fink, Yossi Matias, Idan Szpektor, Craig Boutilier, et al. Dynamic planning in open-ended dialogue using reinforcement learning. *arXiv preprint arXiv:2208.02294*, 2022. URL https://arxiv.org/abs/2208.02294.

[14] Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy Q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/c2073ffa77b5357a498057413bb09d3a-Abstract.html.

[15] Qingfeng Lan, Yangchen Pan, Alona Fyshe, and Martha White. Maxmin Q-learning: Controlling the estimation bias of Q-learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Bkg0u3Etwr.

[16] Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *European Conference on Machine Learning*, pages 282–293. Springer, 2006. URL https://link.springer.com/chapter/10.1007/11871842_29.

[17] Rémi Coulom. Efficient selectivity and backup operators in Monte-Carlo tree search. In *International Conference on Computers and Games*, pages 72–83. Springer, 2007. URL https://link.springer.com/chapter/10.1007/978-3-540-75538-8_7.

[18] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017. URL https://www.nature.com/articles/nature24270.

[19] Ofir Nachum, Shixiang (Shane) Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/e6384711491713d29bc63fc5eeb5ba4f-Paper.pdf.

[20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. URL https://www.nature.com/articles/nature14236.

[21] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *International Conference on Learning Representations*, 2016. URL http://arxiv.org/abs/1511.05952.

[22] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–52, 2015. URL https://dl.acm.org/doi/abs/10.1145/2766462.2767755.

[23] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm Transactions on Interactive Intelligent Systems (TIIS)*, 5(4):1–19, 2015. URL https://dl.acm.org/doi/abs/10.1145/2827872.

[24] Sen Zhao, Wei Wei, Xian-Ling Mao, Shuai Zhu, Minghui Yang, Zujie Wen, Dangyang Chen, and Feida Zhu. Multi-view hypergraph contrastive policy learning for conversational recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 654–664, 2023. URL https://dl.acm.org/doi/abs/10.1145/3539618.3591737.

[25] Mingjie Qian, Yongsen Zheng, Jinghui Qin, and Liang Lin. HutCRS: Hierarchical user-interest tracking for conversational recommender system. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10281–10290. Association for Computational Linguistics, 2023. URL https://aclanthology.org/2023.emnlp-main.635.

[26] Jiarui Jin, Xianyu Chen, Fanghua Ye, Mengyue Yang, Yue Feng, Weinan Zhang, Yong Yu, and Jun Wang. Lending interaction wings to recommender systems with conversational agents. In *Advances in Neural Information Processing Systems*, volume 36, pages 27951–27979, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/58cd3b02902d79aea4b3b603fb0d0941-Paper-Conference.pdf.

[27] Haokun Chen, Xinyi Dai, Han Cai, Weinan Zhang, Xuejian Wang, Ruiming Tang, Yuzhou Zhang, and Yong Yu. Large-scale interactive recommendation with tree-structured policy gradient. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, 2019. URL https://doi.org/10.1609/aaai.v33i01.33013312.

[28] Ali Montazeralghaem, James Allan, and Philip S. Thomas. Large-scale interactive conversational recommendation system using actor-critic framework. In *Proceedings of the 15th ACM Conference on Recommender Systems*, page 220–229, 2021. doi: 10.1145/3460231.3474271. URL https://doi.org/10.1145/3460231.3474271.

[29] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. URL https://www.nature.com/articles/nature16961.

[30] Xiao Yu, Maximillian Chen, and Zhou Yu. Prompt-based Monte-Carlo tree search for goal-oriented dialogue policy planning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7101–7125, 2023. URL https://aclanthology.org/2023.emnlp-main.439.

[31] Tao He, Lizi Liao, Yixin Cao, Yuanxing Liu, Ming Liu, Zerui Chen, and Bing Qin. Planning like human: A dual-process framework for dialogue planning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4768–4791, 2024. URL https://aclanthology.org/2024.acl-long.262.

[32] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double Q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, volume 30, page 2094–2100, 2016. URL https://ojs.aaai.org/index.php/AAAI/article/view/10295.

[33] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

[34] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=F72ximsx7C1.

[35] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2016. URL https://openreview.net/forum?id=SJU4ayYgl.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[37] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48, pages 1995–2003. PMLR, 2016. URL https://proceedings.mlr.press/v48/wangf16.html.

[38] Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3:9–44, 1988. URL https://link.springer.com/article/10.1007/bf00115009.

[39] R Duncan Luce. *Individual choice behavior*, volume 4. Wiley New York, 1959.

[40] Robin L Plackett. The analysis of permutations. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 24(2):193–202, 1975.

[41] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, pages 507–517, 2016. URL https://dl.acm.org/doi/abs/10.1145/2872427.2883037.

[42] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. KGAT: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 950–958, 2019. URL https://dl.acm.org/doi/abs/10.1145/3292500.3330989.

[43] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

[44] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, volume 26, 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/file/1cecc7a77928ca8133fa24680a88d2f9-Paper.pdf.

[45] Xu Han, Shulin Cao, Xin Lv, Yankai Lin, Zhiyuan Liu, Maosong Sun, and Juanzi Li. OpenKE: An open toolkit for knowledge embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 139–144, 2018. URL https://aclanthology.org/D18-2024/.

[46] Guillaume MJ B Chaslot, Mark HM Winands, and H Jaap van Den Herik. Parallel Monte-Carlo tree search. In *Computers and Games: 6th International Conference, CG 2008, Beijing, China, September 29-October 1, 2008. Proceedings 6*, pages 60–71. Springer, 2008. URL https://link.springer.com/chapter/10.1007/978-3-540-87608-3_6.

[47] Aijun Bai, Siddharth Srivastava, and Stuart Russell. Markovian state and action abstractions for MDPs via hierarchical MCTS. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, page 3029–3037, 2016. URL https://www.ijcai.org/Proceedings/16/Papers/430.pdf.

[48] Shuo Zhang and Krisztian Balog. Evaluating conversational recommender systems via user simulation. In *Proceedings of the 26th acm sigkdd international conference on knowledge discovery & data mining*, pages 1512–1520, 2020. URL https://doi.org/10.1145/3394486.3403202.

[49] Tianshu Shen, Jiaru Li, Mohamed Reda Bouadjenek, Zheda Mai, and Scott Sanner. Unintended bias in language model-driven conversational recommendation. *arXiv preprint arXiv:2201.06224*, 2022. URL https://arxiv.org/abs/2201.06224.

[50] Zuohui Fu, Yikun Xian, Shijie Geng, Gerard De Melo, and Yongfeng Zhang. Popcorn: Human-in-the-loop popularity debiasing in conversational recommender systems. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 494–503, 2021. URL https://dl.acm.org/doi/abs/10.1145/3459637.3482461.

[51] Allen Lin, Jianling Wang, Ziwei Zhu, and James Caverlee. Quantifying and mitigating popularity bias in conversational recommender systems. In *Proceedings of the 31st ACM international conference on information & knowledge management*, pages 1238–1247, 2022. URL https://doi.org/10.1145/3511808.3557423.

A An illustration of Conversational Search Tree

An illustration of the conversational search tree is in Figure 2.

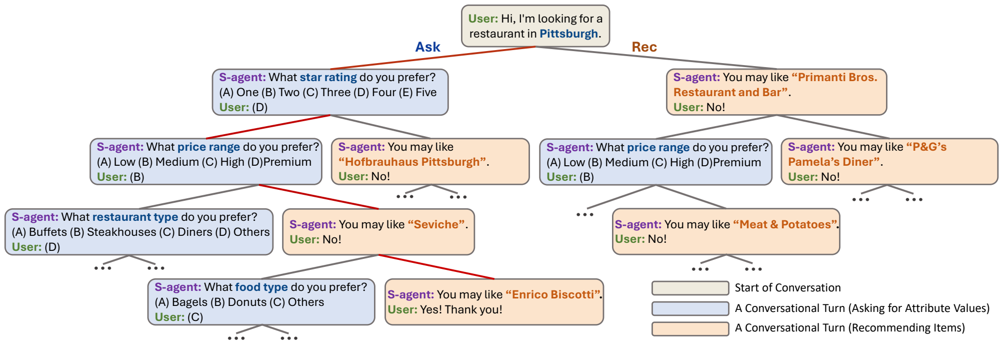

Figure 2: An example of conversational tree search for a user. Conversation starts at the root node with the user specifying preference on an attribute type and its corresponding value. A conversational turn consists of the CRS taking an action and the user responding to that action. The search tree expands as different action types—ask and recommend—are made at each turn. Red line connects the highest-rewarded conversation plan found by the tree.

Table 1: Table of notations.

Notation	Description
$\mathcal{U}, \mathcal{V}, \mathcal{Y}, \mathcal{P}$	the set of users, items, attribute types and attribute values
u, v, y, p	the index of user, item, attribute type and attribute value
t, T	the index of the current turn and the final turn of the conversation
$\mathcal{G}$	the global information graph
$\mathcal{G}_t^+, \mathcal{G}_t^-$	the user's positive feedback graph and negative feedback graph at the t-th turn
$\mathcal{M}(u)$	the MDP environment for user u
$\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma$	the state, action, transition, reward and discount factor in MDP
s_t, o_t, a_t, r_t	the state, action type, action and reward at the t-th turn
$\mathbf{s}_t, \mathbf{a}_t$	the representation of the state s_t, the embedding of the action a_t
$\mathcal{A}_{s_t}$	The action space at the state s_t
$\mathcal{A}_{s_t, o_t}$	The sub action space at the state s_t after choosing the action type o_t
$\mathcal{P}_t^+, \mathcal{P}_t^-, \mathcal{V}_t^-$	the accepted attribute values, the rejected attribute values and the rejected items at the t-th turn
$\mathcal{P}_t^c, \mathcal{V}_t^c$	the candidate attribute values and the candidate items at the t-th turn
$\pi_\phi(o_t \| s_t)$	the policy network that decides the action type $o_t \in \{\texttt{ask}, \texttt{rec}\}$ from the current state s_t
$Q_\theta(a_t \| s_t, o_t)$	the Q-network that decides the specific action a_t according to the action type o_t
$q(s_t, o_t)$	the expected future reward of selecting action type o_t at the state s_t
$\tau_i^{(u)}$	the trajectory from the i-th simulation for the user u
$R(\tau_i^{(u)}, t)$	the cumulative reward of trajectory $\tau_i^{(u)}$ from turn t to the final turn T
$V(s_t)$	the visit count of node s_t during MCTS simulations
$f(s_t, o_t)$	the child node of s_t after choosing the action type o_t
E	training steps
N	the number of simulations in MCTS
w	the exploration factor in UCT

B Notations and Definitions

We denote $\mathcal{U}$ as the set of users, $\mathcal{V}$ as the set of items, $\mathcal{Y}$ as the set of attribute types (e.g., price range, star rating), and $\mathcal{P}$ as the set of attribute values (e.g., medium price range, five-star rating). Each user $u \in \mathcal{U}$ has an interaction history (e.g., view, purchase) with a set of items $\mathcal{V}(u)$. Each item $v \in \mathcal{V}$ is associated with a set of attribute values $\mathcal{P}(v)$, and each attribute type $y \in \mathcal{Y}$ corresponds to a set of attribute values $\mathcal{P}(y)$. Each conversation is initialized by a user specifying preference on an attribute type $y_0 \in \mathcal{Y}$ and its corresponding attribute value $p_0 \in \mathcal{P}$ (e.g., the user says "I am looking for a place with medium price range."). At the t-th conversational turn, S-agent can either choose to ask for preferences over attribute values from a set of candidate attribute values $\mathcal{P}_t^c$, or recommend items from a set of candidate items $\mathcal{V}_t^c$. Based on the user's reply (accept/reject attribute values/items), S-agent repeatedly communicates with users until the user accepts at least one recommended item at turn T (success), or the conversation reaches the maximum number of turns and terminates at $T = T_{\max}$ (fail). *The goal of MCR in general is to successfully recommend at least one item that the user prefers, and complete the conversation in as few turns as possible so as to prevent the user from becoming impatient after too many turns.* Notations in this paper are summarized in Table 1.

C Related Works

Conversational Recommender System CRS understands user preferences through interactive natural language conversations to provide personalized recommendations [7, 8]. Early methods [1, 3] ask users about their desired attribute values to narrow down the list of candidate items to recommend, but are limited under the single-turn setting, as they can only recommend once in a conversation. To address this, multi-turn CRSs allow for multiple turns of question inquiries and item recommendations. For example, EAR [2] adjusts the conversation strategy based on the user's feedbacks with a three-staged process. SCPR [6] models MCR as a path reasoning problem over the knowledge graph of users, items, and attribute values. UNICORN [4] introduces a graph-based RL framework for MCR. MCMIPL [11] develops a multi-interest policy learning framework to understand user's interests over multiple attribute values. HutCRS [25] introduces a user interest tracing module to track user preferences. CORE [26] designs a CRS framework powered by large language models with user-friendly prompts and interactive feedback mechanisms. Chen et al. [27] and Montazeralghaem et al. [28] build a tree-structured index with clustering algorithms to handle the large scale of items and attribute values in MCR. *Different from these methods,* SAPIENT *is able to iteratively improve its planning ability through self-training on demonstrations from MCTS, allowing for more informed and non-myopic conversation strategies.*

Reinforcement Learning for CRS Reinforcement Learning (RL) has achieved great success in tasks requiring strategic planning in complex and interactive environments, such as computer Go [29, 18] and dialogue planning [30, 31]. RL is also employed to train CRS agents to make strategic actions, and current RL-based CRS can be mainly categorized into two types of methods: **(1)** policy-based methods, which train a policy network that directly outputs the probability of taking each action [3, 9], and **(2)** value-based methods, which train a Q-network [32] to estimate the Q-value of actions [4, 11]. Despite promising, these methods may suffer from myopic conversation planning and suboptimal decisions due to their greedy action selection and sampling strategy. *In contrast to these methods, our new* SAPIENT *is able to achieve strategic and non-myopic conversation planning through an MCTS-based planning and self-training algorithm.*

D Transition Function

Transition occurs from the current state s_t to the next state s_{t+1} when the user u responds to the action a_t. The candidate items and attribute values are updated according to the user's response. When the action is to ask, we denote $\widehat{\mathcal{P}}_t^+$ and $\widehat{\mathcal{P}}_t^-$ as the attribute values that the user accepts or rejects at the current turn, and the state is updated as $\mathcal{P}_{t+1}^c = \mathcal{P}_t^c \setminus (\widehat{\mathcal{P}}_t^+ \cup \widehat{\mathcal{P}}_t^-)$, $\mathcal{P}_{t+1}^- = \mathcal{P}_t^- \cup \widehat{\mathcal{P}}_t^-$ and $\mathcal{P}_{t+1}^+ = \mathcal{P}_t^+ \cup \widehat{\mathcal{P}}_t^+$. When the action is to recommend, if the user rejects the recommended items, we denote $\widehat{\mathcal{V}}_t^-$ as the recommended items that have all been rejected at the current turn and update the state as $\mathcal{V}_{t+1}^- = \mathcal{V}_t^- \cup \widehat{\mathcal{V}}_t^-$; otherwise the conversation terminates since the user has accepted a recommended item. Finally, we update the candidate item set to include only those items that remain not rejected and whose attribute values share common traits with the accepted attribute values: $\mathcal{V}_{t+1}^c = \{v | v \in \mathcal{V}_{p_0} \setminus \mathcal{V}_{t+1}^- \text{ and } \mathcal{P}(v) \cap \mathcal{P}_{t+1}^+ \neq \emptyset \text{ and } \mathcal{P}(v) \cap \mathcal{P}_{t+1}^- = \emptyset\}$, where $\mathcal{V}_{p_0}$ denotes the set of items that are associated with the attribute value p_0 specified at the start of the conversation.

E Reward Function

Following Lei et al. [9], Zhang et al. [11], for different conversation scenarios, we consider five kinds of immediate rewards at given conversation turn: (1) $r_{\text{rec}}^+ = 1$: a large positive value when the user accepts a recommended item; (2) $r_{\text{ask}}^+ = 0.01$: a small positive value when the user accepts an attribute value asked by the CRS; (3) $r_{\text{rec}}^- = -0.1$: a negative value when the user rejects a recommended item; (4) $r_{\text{ask}}^{-} = -0.1$: a negative value when the user rejects an attribute value asked by the CRS; and (5) $r_{\text{quit}} = -0.3$: a large negative value if the conversation reaches the maximum number of turns. In addition, since our system follows the multi-choice CRS setting, we sum up the positive and negative rewards for multiple attribute values specified in a multiple-choice question: $r_t = \sum_{\widehat{\mathcal{P}}_t^+} r_{\text{ask}}^{+} + \sum_{\widehat{\mathcal{P}}_t^-} r_{\text{ask}}^{-}$.

F Details of the State Encoder

Global information graph encoder captures the global relationships between similar users and items, as well as the correlations between items and attribute values from the global information graph $\mathcal{G}$. We build $\mathcal{G}$ with the following rules: an edge $e_{u,v} \in \mathcal{E}_{\mathcal{U},\mathcal{V}}$ exists between a user u and an item v iff. the user u has interacted with item v, and an edge $e_{p,v} \in \mathcal{E}_{\mathcal{P},\mathcal{V}}$ exists between an attribute value p and an item v iff. item v is associated with attribute value v. Next, let $\mathbf{h}_u^{(0)} = \mathbf{e}_u$, $\mathbf{h}_v^{(0)} = \mathbf{e}_v$ and $\mathbf{h}_p^{(0)} = \mathbf{e}_p$ denote the embeddings of user, item and attribute value, we adopt a multi-head Graph Attention Network (GAT) [33, 34] to iteratively refine the node embeddings with neighbourhood information:

$$\mathbf{z}_i^{(l+1)} = \Big\Vert_{k=1}^{K} \sum_{j\in\mathcal{N}_i} \alpha_{ij}^k \mathbf{W}_{2,k}^{(l)} \mathbf{h}_j \quad \alpha_{ij}^k = \frac{\exp(\mathbf{a}_k^{(l)\top} \sigma(\mathbf{W}_{1,k}^{(l)}\mathbf{h}_i + \mathbf{W}_{2,k}^{(l)}\mathbf{h}_j))}{\sum\limits_{j'\in\mathcal{N}_i} \exp(\mathbf{a}_k^{(l)\top} \sigma(\mathbf{W}_{1,k}^{(l)}\mathbf{h}_i + \mathbf{W}_{2,k}^{(l)}\mathbf{h}_{j'}))}$$

$$\mathcal{N}_i = \begin{cases} \{v \,|\, e_{i,v} \in \mathcal{E}_{\mathcal{U},\mathcal{V}}\}, & \text{if} \quad i \in \mathcal{U} \\ \{v \,|\, e_{i,v} \in \mathcal{E}_{\mathcal{P},\mathcal{V}}\}, & \text{if} \quad i \in \mathcal{P} \\ \{u \,|\, e_{u,i} \in \mathcal{E}_{\mathcal{U},\mathcal{V}}\}, & \text{if} \quad i \in \mathcal{V}, \mathcal{N}_i \subset \mathcal{U} \\ \{p \,|\, e_{p,i} \in \mathcal{E}_{\mathcal{P},\mathcal{V}}\}, & \text{if} \quad i \in \mathcal{V}, \mathcal{N}_i \subset \mathcal{P} \end{cases}, \tag{1}$$

where $\mathbf{a}_k^{(l)} \in \mathbb{R}^{d/K}$, $\mathbf{W}_{1,k}^{(l)} \in \mathbb{R}^{(d/K)\times d}$,$\mathbf{W}_{2,k}^{(l)} \in \mathbb{R}^{(d/K)\times d}$ are the trainable parameters for the l-th layer, K denotes the number of attention heads, σ denotes the LeakyReLU activation function, $||$ denotes the concatenation operation. For the user and attribute value node, its hidden representation of the l+1-th layer is obtain from $\mathbf{h}_u^{(l+1)} = \sigma(\mathbf{z}_u^{(l+1)})$, $\mathbf{h}_p^{(l+1)} = \sigma(\mathbf{z}_p^{(l+1)})$. While for the item node, its hidden representation of the l+1-th layer is obtained by aggregating the information from both its neighbourhood users and attribute values: $\mathbf{h}_v^{(l+1)} = \sigma((\mathbf{z}_{v,\mathcal{N}_v\subset\mathcal{U}}^{(l+1)} + \mathbf{z}_{v,\mathcal{N}_v\subset\mathcal{P}}^{(l+1)})/2)$. We stack L_g layers of GATs and fetch the hidden representations $\mathbf{h}_u^{(L_g)}$, $\mathbf{h}_v^{(L_g)}$, $\mathbf{h}_p^{(L_g)}$ at the last layer as the output of the global information graph encoder.

Positive feedback graph encoder captures the user's positive feedback on attribute values and their relations with candidate attribute values/items in the conversation history. For each user u, at the t-th conversational turn, we construct a local positive graph $\mathcal{G}_t^+ =< (\{u\} \cup \mathcal{P}_t^+ \cup \mathcal{P}_t^c \cup \mathcal{V}_t^c), \mathcal{E}_t^+ >$, where the weight of the edge $\mathcal{E}_t^+(i,j)$ between node i and j is constructed from the following rules:

$$\mathcal{E}_t^+(i,j) = \begin{cases} w_v^{(t)}, & \text{if} \quad i \in \mathcal{U}, j \in \mathcal{V} \\ 1, & \text{if} \quad i \in \mathcal{V}, j \in \mathcal{P} \\ 1, & \text{if} \quad i \in \mathcal{U}, j \in \mathcal{P}_t^+ \\ 0, & \text{otherwise} \end{cases}, \tag{2}$$

where $w_v^{(t)} = \text{sigmoid}(\mathbf{e}_u^\top \mathbf{e}_v + \sum_{p\in\mathcal{P}_t^+} \mathbf{e}_v^\top \mathbf{e}_p - \sum_{p\in\mathcal{P}_t^-} \mathbf{e}_v^\top \mathbf{e}_p)$ denotes the dynamic matching score of the item v at the current conversational turn t. Next, let $\mathbf{e}_u^{(0)} = \mathbf{e}_u$, $\mathbf{e}_v^{(0)} = \mathbf{e}_v$ and $\mathbf{e}_p^{(0)} = \mathbf{e}_p$ denote the embeddings of user, item and attribute value, we then adopt a Graph Convolutional Network (GCN) [35] to propagate the message on the current dynamic graph, and calculate the hidden representation of the node at the l+1-th layer as follows:

$$\mathbf{e}_i^{(l+1)} = \sigma\Big(\sum_{\{j|\mathcal{E}_t^+(i,j)>0\}} \frac{\mathbf{W}_a^{(l)}\mathbf{e}_j^{(l)}}{\sqrt{\sum_{\hat{j}}\mathcal{E}_t^+(i,\hat{j})\sum_{\hat{j}}\mathcal{E}_t^+(j,\hat{j})}} + \mathbf{e}_i^{(l)}\Big), \tag{3}$$

where $\mathbf{W}_a^{(l)} \in \mathbb{R}^{d\times d}$ are the trainable parameters for the l-th layer, σ denotes the LeakyReLU activation function. We stack L_a layers of GCNs and fetch the hidden representation $\mathbf{e}_u^{(L_a)}$, $\mathbf{e}_v^{(L_a)}$,$\mathbf{e}_p^{(L_a)}$ at the last layer as the output of the positive feedback graph encoder.

Negative feedback graph encoder captures the user's negative feedback on attribute values and their negative correlations with candidate attribute values/items in the conversation history. Similar to Eq. 2, for each user u, we construct a local negative graph $\mathcal{G}_t^- =< \{u\} \cup \mathcal{P}_t^- \cup \mathcal{V}_t^- \cup \mathcal{P}_t^c \cup \mathcal{V}_t^c, \mathcal{E}_t^- >$, where the weight of the edge $\mathcal{E}_{i,j}^{(t)}$ between node i and node j is constructed from the following rules:

$$\mathcal{E}_t^-(i,j) = \begin{cases} w_v^{(t)}, & \text{if} \quad i \in \mathcal{U}, j \in \mathcal{V} \\ 1, & \text{if} \quad i \in \mathcal{V}, j \in \mathcal{P} \\ 1, & \text{if} \quad i \in \mathcal{U}, j \in \mathcal{P}_t^- \\ 0, & \text{otherwise} \end{cases} . \tag{4}$$

We then stack L_n layers of GCNs similar to Eq. 3, and fetch the hidden representation $\mathbf{e}_u^{(L_n)}$, $\mathbf{e}_v^{(L_n)}$,$\mathbf{e}_p^{(L_n)}$ at the last layer as the output of the negative feedback graph encoder.

Transformer-based aggregator fuses the information from the graph encoders, and captures the sequential relationships among items and attribute values mentioned in the conversation history. Specifically, for the accepted/rejected attribute values/items at previous conversational turns, we first project the accepted ones and rejected ones into different spaces to distinguish between the positive and the negative feedbacks:

$$\begin{aligned} \mathbf{e}_p^{'} = \mathbf{W}_a \mathbf{e}_p^{(L_a)} + \mathbf{b}_a \text{ or } \mathbf{e}_p^{'} = \mathbf{W}_n \mathbf{e}_p^{(L_n)} + \mathbf{b}_n \\ \mathbf{e}_v^{'} = \mathbf{W}_n \mathbf{e}_v^{(L_n)} + \mathbf{b}_n, \end{aligned} \tag{5}$$

where $\mathbf{W}_a, \mathbf{W}_n \in \mathbb{R}^{d \times d}$ and $\mathbf{b}_a, \mathbf{b}_n \in \mathbb{R}^d$ are trainable parameters. Next, the positive/negative feedbacks are fused with the representations from the global graph encoder with a gating mechanisms to capture the information from both the global relationships and the local conversation feedbacks:

$$\begin{aligned} \mathbf{v}_p = \text{gate}(\mathbf{h}_p^{(L_g)}, \mathbf{e}_p^{'}),\ \mathbf{v}_v = \text{gate}(\mathbf{h}_v^{(L_g)}, \mathbf{e}_v^{'}) \\ \text{gate}(\mathbf{x}, \mathbf{y}) = \xi \cdot \mathbf{x} + (1 - \xi) \cdot \mathbf{y} \\ \xi = \text{sigmoid}(\mathbf{W}_1^{ga} \mathbf{x} + \mathbf{W}_2^{ga} \mathbf{y} + \mathbf{b}^{ga}), \end{aligned} \tag{6}$$

where $\mathbf{W}_1^{ga}, \mathbf{W}_2^{ga} \in \mathbb{R}^{d \times d}$, $\mathbf{b}^{ga} \in \mathbb{R}^d$ are trainable parameters. Finally, we adopt a Transformer encoder [36] to capture the sequential relationships about the conversation history and obtain the current state representation $\mathbf{s}_t$ as follows:

$$\mathbf{s}_t = \text{Meanpooling}(\text{Transformer}(\mathbf{V})), \tag{7}$$

where $\mathbf{V}$ is built using all the previously mentioned attribute values and items and in the conversation history: $\mathbf{V} = \{\mathbf{v}_p | p \in \mathcal{P}_t^+ \cup \mathcal{P}_t^-\} \cup \{\mathbf{v}_v | v \in \mathcal{V}_t^-\}$.

G Details of the Policy Network and Q-network

Based on the state representation $\mathbf{s}_t$, S-agent adopts a policy network $\pi_\phi(o_t|s_t)$ to decide action type o_t and a Q-network $Q_\theta(a_t|s_t, o_t)$ to decide the specific action a_t:

$$\pi_\phi(o_t|s_t) = \text{softmax}(\text{MLP}_\pi(\mathbf{s}_t)) \quad Q_\theta(a_t|s_t, o_t) = \text{MLP}_A(\mathbf{s}_t||\mathbf{a}_t) + \text{MLP}_V(\mathbf{s}_t), \tag{8}$$

where MLP denotes a two-layer perceptron, $\mathbf{a}_t = \mathbf{e}_p$ or $\mathbf{a}_t = \mathbf{e}_v$ denotes the embedding of actions (attribute value or item) at the t-th turn, A and V denote the advantage and value function of the dueling Q-network [37] respectively.

H Training Algorithm

Training algorithm of `SAPIENT` is in Algorithm 1.

I Details of Building the Search Tree

The search tree in S-planner is built in the following four stages:

Algorithm 1 Training algorithm of `SAPIENT`

Require: conversational MDPs for all users $\{\mathcal{M}(u)\}_{u=1}^{\mathcal{U}}$, training steps E, # of simulations N, exploration factor w

for step $\leftarrow 1, \cdots, E$ **do**
  Sample a user u from $\mathcal{U}$, initialize the state as s_0
  for $n \leftarrow 1, \cdots, N$ **do**
    Initialize the trajectory as $\tau_i^{(u)} \leftarrow \{\}, t \leftarrow 0$
    while s_t has children **do** ▷ Trajectory Selection
      Select an action type o_t (Eq. 9) and an action a_t (Eq. 10)
      Save s_t to $\tau_i^{(u)}$, $s_{t+1} \leftarrow \mathcal{T}(s_t, a_t), t{+}{=}1$
    end while
    while s_t is not end of conversation **do**
      Attach two children (`ask` and `rec`) to s_t ▷ Node Expansion
      Select an action type o_t with policy network ▷ Conversation Simulation
      Select an action a_t with the Q-network
      Save s_t to $\tau_i^{(u)}$, $s_{t+1} \leftarrow \mathcal{T}(s_t, a_t), t{+}{=}1$
    end while
    Initialize $R(\tau_i^{(u)}, t) \leftarrow r_t$
    while $t \geq 0$ **do** ▷ Reward Back-Propagation
      Update the Q-value $q(s_t, o_t)$ with Eq. 11
      $R_t(\tau_i^{(u)}) \leftarrow \gamma R_t(\tau_i^{(u)}) + r_t, V(s_t){+}{=}1, t{-}{=}1$
    end while
  end for
  Save the highest-rewarded trajectory to the memory ▷ Training
  Sample a batch of experiences, update $\pi_\phi(o_t|s_t)$ and $Q_\theta(a_t|s_t, o_t)$
end for

Trajectory selection S-planner selects the most promising trajectory from the tree root to a leaf node that is likely to obtain a high future reward $q(s_t, o_t)$, and the selected trajectory will be further expanded for conversation simulation later. This selection process trades off between exploitation, measured by $q(s_t, o_t)$, against exploration, measured by how often the nodes are visited. Particularly, S-planner adapts the Upper Confidence bounds applied to Trees (UCT) approach [16] to achieve the trade-off between exploitation and exploration, and at each node s_t, select action type o_t^* into the trajectory that maximizes the UCT value as follows:

$$o_t^* \leftarrow \underset{o_t \in \{\texttt{ask},\texttt{rec}\}}{\arg\max} \left[q(s_t, o_t) + w\sqrt{\frac{\log V(s_t)}{V(f(s_t, o_t))}} \right], \tag{9}$$

where $w > 0$ is the exploration factor, $V(s_t)$ quantifies the visits on node s_t during conversation simulations, and $f(s_t, o_t)$ represents the child node of s_t after choosing the action type o_t. Intuitively, the second term in Equation 9 is larger if the child node is less visited, encouraging more exploration. After selecting the action type o_t^*, S-planner chooses the optimal action a_t^* with the Q-network:

$$a_t^* \leftarrow \underset{a_t \in \mathcal{A}_{s_t, o_t^*}}{\arg\max}\, Q_\theta(a_t|s_t, o_t^*). \tag{10}$$

Node Expansion When a leaf node is reached, S-planner expands the leaf node by attaching two children node (corresponding to two action type `ask` and `rec`) to it. The expected future reward $q(s_{t+1}, o_{t+1})$ of choosing o_{t+1} at the newly attached node s_{t+1} is initialized as the highest value estimated by the Q-network $Q_\theta(a_{t+1}|s_{t+1}, o_{t+1})$ among all the candidate actions in the sub action space $\mathcal{A}_{s_{t+1}, o_{t+1}}$, serving as a heuristic guidance for the future tree search.

Conversation Simulation To predict how the future conversation unfolds, S-planner continues to simulate conversations between `SAPIENT` and the user until the conversation succeeds or terminates. Starting from the last expanded node, at each turn, the policy network decides the action type, while the Q-network decides the specific action.

Reward Back-propagation Once the simulated conversation succeeds or terminates, S-planner back-propagates from the leaf node of the current trajectory $\tau_i^{(u)}$ to the root to increase the visit count of each node along $\tau_i^{(u)}$, and update the expected future reward $q(s_t, o_t)$ along $\tau_i^{(u)}$ as follows:

$$q(s_t, o_t) \leftarrow q(s_t, o_t) + \Big[R_t(\tau_i^{(u)}) - q(s_t, o_t)\Big] / V(s_t), \tag{11}$$

where $R_t(\tau_i^{(u)})$ denotes the cumulative reward up until the t-th turn. Intuitively, this update rule is similar to stochastic gradient ascent: each time $q(s_t, o_t)$ is adjusted by step $1/V(s_t)$ in the direction of the error $R_t(\tau_i^{(u)}) - q(s_t, o_t)$.

J Details on S-agent training

S-agent is trained with the best conversation plan from S-planner to improve its planning capability. The policy network in S-agent is updated with supervised cross-entropy loss, and the Q-network is updated with double Q-learning [32]. Also, an efficient algorithm SAPIENT-e is proposed to improve the training efficiency. Details are as follows:

Policy Network Update The policy network is updated with the supervised cross-entropy loss, aligning its decision with MCTS demonstrations:

$$\mathcal{L}_\phi = \mathbb{E}_{e_t \sim \mathcal{D}} \left[-\log \pi_\phi(o_t|s_t) \right]. \tag{12}$$

Q-Network Update The Q-network is updated with double Q-learning [32], which maintains a target network $Q_{\tilde{\theta}}(a_t|s_t, o_t)$ as a periodic copy of the online network $Q_\theta(a_t|s_t, o_t)$ and trains $Q_\theta(a_t|s_t, o_t)$ to minimize the temporal difference error [38]:

$$\mathcal{L}_\theta = \mathbb{E}_{e_t \sim \mathcal{D}} \Big[(Q_\theta(a_t|s_t, o_t) - r_t - \gamma \max_{a_{t+1} \in \mathcal{A}_{s_{t+1}, o_{t+1}}} Q_{\tilde{\theta}}(a_{t+1}|s_{t+1}, o_{t+1}))^2 \Big], \tag{13}$$

where γ is the discount factor in MDP, $\tilde{\theta}$ is the parameters of the target network.

Improving Training Efficiency The aforementioned training process only selects the highest-rewarded conversation plan as demonstrations. This guarantees the quality of training data, but may be inefficient because it requires simulating a large number of conversations to collect enough highest-rewarded trajectories. To improve efficiency, we further propose a variant SAPIENT-e. Instead of using only the highest-rewarded trajectories, SAPIENT-e makes full use of all the trajectories found by S-planner, and thus greatly reduces the cost of collecting enough trajectories. Since some trajectories are good while others are suboptimal (e.g., user quits the conversation after $T_{\max}$ turns), we would like to encourage $\pi_\theta(o_t|s_t)$ to increase the likelihood for good trajectories and decrease the likelihood for suboptimal ones. To this end, we employ the Plackett-Luce model [39, 40] to train $\pi_\theta(o_t|s_t)$ with listwise likelihood estimations. For each user u, assuming that all the N trajectories are ranked by their cumulative rewards in the order of $\tau_1^{(u)}, \tau_2^{(u)}, \cdots, \tau_N^{(u)}$, we update the policy network as:

$$\mathcal{L}_\phi = \mathbb{E}_{u \sim \mathcal{U}} \Big[-\log P(\tau_1^{(u)} \succ \tau_2^{(u)} \succ \cdots \succ \tau_N^{(u)}) \Big] = \mathbb{E}_{u \sim \mathcal{U}} \left[-\log \prod_{n=1}^{N} \frac{\exp \sum_{s_t, o_t \in \tau_n^{(u)}} \log \pi_\phi(o_t|s_t)}{\sum_{j=n}^{N} \exp \sum_{s_t, o_t \in \tau_j^{(u)}} \log \pi_\phi(o_t|s_t)} \right], \tag{14}$$

where $\tau_1^{(u)} \succ \tau_2^{(u)}$ indicates $\tau_1^{(u)}$ is preferred over $\tau_2^{(u)}$, and the denominator sums the likelihood of all the trajectories preferred over the j-th trajectory. The Q-network is still updated as in Equation 13 except that the sampled experiences come from all the trajectories instead of only the highest-rewarded trajectories. In this way, all the trajectories found by S-planner can be utilized, thus saving the search cost. SAPIENT-e only performs slightly worse than SAPIENT and much better than baselines (Section 4.1), and can be viewed as a good trade-off between efficiency and performance.

K Experimental Details

K.1 Dataset Details and Statistics

We evaluate our method on four benchmark datasets: Yelp [2], LastFM [2], Amazon-Book [22, 41] and MovieLens [23]. Detailed dataset information is introduced as follows and the statistics are presented in Table 2.

- **Yelp**[1]: This dataset contains the users' reviews on business venues such as restaurants and bars. Lei et al. [2] build a 2-layer taxonomy for the original attribute values in this dataset, and we adopt the first-layer categories as attribute types, and the second-layer categories as attribute values.
- **LastFM**[2]: This dataset contains music artist listening information of users from an online music system. Following Lei et al. [9], Zhang et al. [11], we adopt cluster the original attribute values into 34 attribute types.
- **Amazon-Book**[3]: The Amazon review dataset [22, 41] is a large-scale collection of online shopping data featuring users' product reviews across various domains. We select the book domain from this collection. Following Wang et al. [42] we choose entities and relations within knowledge graph as attribute values and types, and only retain entities associated with at least 10 items to ensure dataset quality.
- **MovieLens**[4][23]: This dataset records users' activities in an online movie recommendation platform. We select the version with about 20M interactions, choose entities and relations within knowledge graph as attribute values, and only retain the user-item interactions ratings greater than 3 to ensure dataset quality.

Table 2: Statistics of datasets after preprocessing.

Dataset	Yelp	LastFM	Amazon-Book	MovieLens
#Users	27,675	1,801	30,291	20,892
#Items	70,311	7,432	17,739	16,482
#Interactions	1,368,609	76,693	478,099	454,011
#Attribute Values	590	8,438	988	1,498
#Attribute types	29	34	40	21
#Entities	98,576	17,671	49,018	38,872
#Relations	3	4	2	2
#Triplets	2,533,827	228,217	565,069	380,016

K.2 Details of the User Simulator

Training and evaluating CRS with real user interactions can be impractically expensive at scale. To address this issue, we follow Zhang et al. [11] and simulate a conversation session for each observed user-item set interaction pair $(u, \mathcal{V}(u))$. In each simulated conversation, we regard an item $v_i \in \mathcal{V}(u)$ as the ground-truth target item. Each conversation is initialized with a user specifying preference on an attribute value p_0 that this user clearly prefers, which is randomly chosen from the shared attribute values of all items in $\mathcal{V}(u)$. As the conversation continues, in each turn, the simulated user feedback follows these rules: (1) when the CRS asks a question, the user will only accept attribute values associated with any item in $\mathcal{V}(u)$ and reject others; (2) when the CRS recommends a list of items, the user will accept it only if at least one item in $\mathcal{V}(u)$ is in the recommendation list; (3) the user will become impatient after $T_{\max} = 15$ turns and quit the conversation.

K.3 Detailed Calculation of hDCG

Normalized Discounted Cumulative Gain@K (NDCG@K) is a common ranking metric to evaluate the relevance of items recommended by a system. Deng et al. [4] extend the normal NDCG to a two-level hierarchical version suitable for evaluating CRS. The hierarchical normalized Discounted

[1] https://www.yelp.com/dataset/
[2] https://grouplens.org/datasets/hetrec-2011/
[3] https://jmcauley.ucsd.edu/data/amazon/
[4] https://grouplens.org/datasets/movielens/

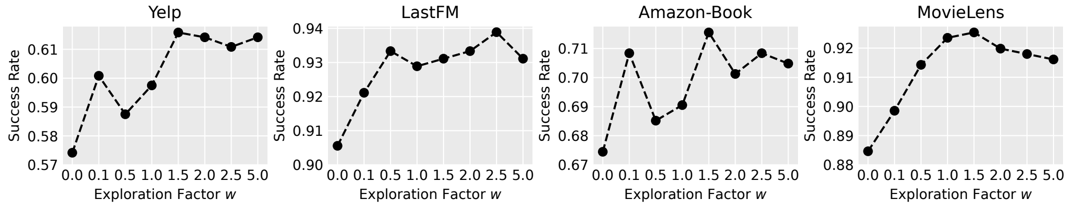

Figure 3: Success rate under different exploration factor w.

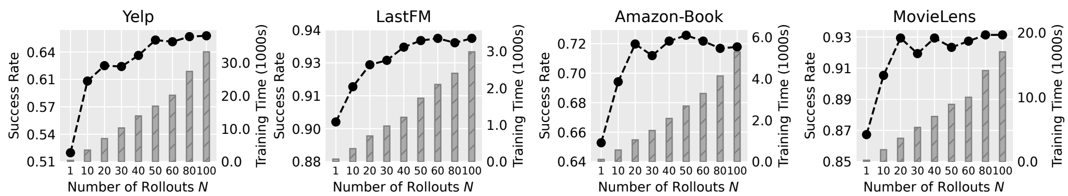

Figure 4: Success rate and training time for 100 gradient descent steps under different rollout number N.

Cumulative Gain@(T, K) (hDCG@(T, K)) is defined as follows:

$$hDCG@(T,K) = \sum_{t=1}^{T}\sum_{k=1}^{K} r(t,k)\left[\frac{1}{\log_2(t+2)} + \left(\frac{1}{\log_2(t+1)} - \frac{1}{\log_2(t+2)}\right)\frac{1}{\log_2(k+1)}\right], \tag{15}$$

where T represents the number of conversation turns, K represents the number of items recommended at each turn, $r(t,k)$ denotes the relevance of the result at turn t and position k. Since we have a maximum of T_{max} conversation turns, and the CRS may recommend a maximum number of K_v items, we report the $hDCG$ metric where $T = T_{\text{max}}$ and $K = K_v$.

K.4 Details of Baseline Methods

For a comprehensive evaluation, we compare our method with the following baselines:

- **Max Entropy** [2]. This method chooses to ask attribute values with the maximum entropy among candidate items, or chooses to recommend the top-ranked items with certain probability.
- **Abs Greedy** [1]. This method only recommends items in each turn without asking questions. If the recommended items are rejected, the model updates by treating them as negative samples.
- **CRM** [3]. This method adopts a policy network to decide when and what to ask. As it is originally designed for single-turn CRS, we follow Lei et al. [2] to adapt it to the MCR setting.
- **EAR** [2]. This method designs a three-stage strategy to better converse with users. It first builds predictive models to estimate user preferences, then learns a policy network to take action, and finally updates the recommendation model with reflection mechanism.
- **SCPR** [6]. This method models CRS as an interactive path reasoning problem over the knowledge graph. It leverages the graph structure to prune irrelevant candidate attribute values and adopts a policy network to choose actions.
- **UNICORN** [4]. This method designs a unified CRS policy learning framework with graph-based state representation learning and deep Q-learning.
- **MCMIPL** [11]. This method develops a multi-choice questions based multi-interest policy learning framework for CRS, which enables users to answer multi-choice questions in attribute combinations.
- **HutCRS** [25]. This method proposes a user interest tracking module integrated with the decision-making process of the CRS to better understand the preferences of the user.
- **CORE** [26]. This is a Large Language Model (LLM) powered CRS chatbot with user-friendly prompts and interactive feedback mechanisms.

K.5 Implementation Details

Following Zhang et al. [11] for a more realistic multi-choice setting, if the CRS agent decides to ask, top-K_p attribute values with the same attribute type will be asked from the candidate attribute value set $\mathcal{P}_t^c$ to form a multi-choice question; and if the CRS agent decides to recommend, top-K_v items will be recommended from the candidate item set $\mathcal{V}_t^c$. Following Lei et al. [6], Deng et al. [4], each dataset is randomly split into train, validation and test by a 7:1.5:1.5 ratio. We set the embedding dimension d as 64, batch size as 128. We adopt an Adam optimizer [43] with a learning rate of 1e-4. We set the discount factor γ as 0.999. The memory size of experience replay is set as 10000. For the GPM module, the number of the collaborative graph encoder layers L_c is set as 2, and both the number of the positive graph encoder layer L_a and the number of the negative graph encoder layer L_n are set as 1, the number of the Transformer-based aggregator layers are set as 2, and we follow Deng et al. [4], Zhang et al. [11] to adopt TranE [44] from OpenKE [45] to pretrain the node embeddings with the training set. Following Lei et al. [2], Deng et al. [4], we set the size of recommendation list K_v as 10, the maximum number of turns $T_{\max} = 15$. We set the default exploration factor w as 1.5, the default number of rollouts N as 20, and variants are explored in Section 4.3.

K.6 Overall Performance Comparison

Table 3: Performances on benchmark datasets. The best performance and the second-best performance on each dataset is in bold and underlined respectively. $*$ indicates the improvement is statistically significant ($p < 0.01$).

Models	Yelp			LastFM			Amazon-Book			MovieLens		
	SR↑	AT↓	hDCG↑	SR↑	AT↓	hDCG↑	SR↑	AT↓	hDCG↑	SR↑	AT↓	hDCG↑
`Abs Greedy`	0.195	14.08	0.069	0.539	10.92	0.251	0.214	13.50	0.092	0.752	4.94	0.481
`Max Entropy`	0.375	12.57	0.139	0.640	9.62	0.288	0.343	12.21	0.125	0.704	6.93	0.448
`CRM`	0.223	13.83	0.073	0.597	10.60	0.269	0.309	12.47	0.117	0.654	7.86	0.413
`EAR`	0.263	13.79	0.098	0.612	9.66	0.276	0.354	12.07	0.132	0.714	6.53	0.457
`SCPR`	0.413	12.45	0.149	0.751	8.52	0.339	0.428	11.50	0.159	0.812	4.03	0.547
`UNICORN`	0.438	12.28	0.151	0.843	7.25	0.363	0.466	11.24	0.170	0.836	3.82	0.576
`MCMIPL`	0.482	11.87	0.160	0.874	6.35	0.396	0.545	10.83	0.223	0.882	3.61	0.599
`HutCRS`	0.528	11.33	0.175	0.900	6.52	0.348	0.638	9.84	0.227	0.902	4.16	0.475
`CORE`	0.210	12.82	0.166	0.862	7.05	0.356	0.462	11.49	0.182	0.810	6.51	0.429
`SAPIENT-e`	0.612*	10.41*	0.208*	0.922*	6.32	0.358	0.682*	9.51*	0.239*	0.928*	3.76	0.607*
`SAPIENT`	**0.622***	**10.02***	**0.229***	**0.928***	**6.15***	**0.398**	**0.718***	**9.28***	**0.252***	**0.930***	**3.48***	**0.610***

We compare `SAPIENT` with 9 state-of-the-art baselines and report the experimental results in Table 3. We have the following observations:

(1) `SAPIENT` ***achieves consistent improvement over baselines in terms of all metrics on all the datasets, with an average improvement of 9.1% (SR), 6.0% (AT) and 11.1% (hDCG) compared with the best baseline.*** Different from baselines, which base their planning solely on the observation of current state without looking ahead, `SAPIENT` foresees how the future conversation unfolds with an MCTS-based planning algorithm. This enables `SAPIENT` to take actions that maximize the cumulative rewards instead of settling for the immediate reward, enabling strategic, non-myopic conversational planning and superior performances.

(2) `SAPIENT` ***substantially outperforms baselines in datasets demanding strong strategic planning capability from the CRS.*** The performance gain of `SAPIENT` is higher on datasets with a larger AT (Yelp and Amazon-Book) compared to datasets with a smaller AT (LastFM and MovieLens), and higher AT in these datasets indicates the need for more strategic planning over long conversational turns. Compared with baselines, `SAPIENT` is equipped with `S`-planner and excels in conversational planning, hence showing remarkable improvements on these two datasets.

(3) `SAPIENT`***-e outperforms all baselines on recommendation success rate.*** Although the training data for `SAPIENT-e` still contain a portion of low-quality trajectories , `SAPIENT-e` still significantly outperforms the best baselines across most metrics, indicating that `SAPIENT-e` is a good trade-off between efficiency and performance.

K.7 Efficiency Analysis

Table 4: Training GPU hours on four datasets.

Model	Yelp	LastFM	Book	MovieLens
UNICORN	16.15	4.30	6.03	7.96
MCMIPL	15.57	5.08	6.40	7.93
HutCRS	14.05	4.66	5.83	8.40
SAPIENT-e	16.40	5.57	6.88	8.45
SAPIENT	38.15	11.07	13.21	20.97

Training efficiency of* SAPIENT *and* SAPIENT*-e is highly comparable to the baselines. As shown in Table 4, SAPIENT-e takes similar training time with baselines as it collects all the trajectories from MCTS and do not incur additional search cost. Even with SAPIENT, the training time is only about 2 times longer than baselines. This is because conversation simulation only requires forward propagation without gradient backward, so even conducting 20 rollouts per user will not significantly reduce efficiency. Also note that during inference, the efficiency of SAPIENT is comparable with baselines, as tree search is not required during inference.

K.8 Additional Analysis on Efficiency

Although training SAPIENT requires conducting multiple simulated rollouts for each user, we find that such design will not significantly compromise efficiency compared with the baseline CRS methods. Under the same training pipeline with a single Tesla V100 GPU, SAPIENT with 20 rollouts per user takes 698 seconds per 100 iteration steps on the LastFM dataset and 1049 seconds per 100 iteration steps on the Amazon-Book dataset on average, which is about twice as slow as the two competitive baselines (HutCRS: 305 seconds/100 steps on LastFM, 465 seconds/100 steps on Amazon-Book; MCMIPL: 429 seconds/100 steps on LastFM, 548 seconds/100 steps on Amazon-Book). This is because the simulation process only requires forward calculation without the need for gradient backward update, so even conducting 20 rollouts per user will only reduce the training speed by half. Also, we note that during inference, the efficiency of SAPIENT is comparable with baseline methods, because no tree search is required during inference. Therefore, we think that it is worthwhile to introduce tree search for CRS, as such design only compromises efficiency during training and can be accelerated by adopting parallel methods for MCTS [46] in the future.

K.9 Ablation Study

Table 5: Experimental results of the ablation study.

Models	Yelp			LastFM			Amazon-Book			MovieLens		
	SR↑	AT↓	hDCG↑	SR↑	AT↓	hDCG↑	SR↑	AT↓	hDCG↑	SR↑	AT↓	hDCG↑
SAPIENT	0.622	10.02	0.229	0.928	6.15	0.398	0.718	9.28	0.252	0.930	3.48	0.610
w/o Global $\mathcal{G}$	0.520	11.39	0.171	0.906	6.56	0.345	0.626	10.15	0.217	0.878	5.30	0.397
w/o Pos. $\mathcal{G}^+$	0.482	11.56	0.163	0.862	7.53	0.313	0.560	11.15	0.184	0.886	4.17	0.496
w/o Neg. $\mathcal{G}^-$	0.532	10.80	0.185	0.905	6.92	0.336	0.656	10.01	0.227	0.860	5.42	0.389
w/o Policy net.	0.519	11.08	0.186	0.894	6.37	0.361	0.628	9.61	0.240	0.896	4.59	0.516
w/o Q-net.	0.582	10.69	0.190	0.808	7.92	0.332	0.594	10.62	0.198	0.866	5.47	0.386
w/o S-planner	0.520	11.06	0.193	0.902	6.80	0.335	0.650	10.20	0.218	0.860	5.53	0.396

To validate the effectiveness of the key components, we conduct ablation studies and report the results in Table 5. Based on the result, we have the following observations.

Effectiveness of state encoding To validate the effectiveness of the key components, we conduct ablation studies and report the results in Table 5. We have the following observations:

(1) ***Each graph—$\mathcal{G}, \mathcal{G}^+, \mathcal{G}^-$ is vital for* S*-agent to encode the state information.*** Removing each graph from S-agent degrades performance, verifying the necessity of each graph in state encoding: global information graph $\mathcal{G}$ is crucial for mining user-item relations and item-attribute value associations, while positive ($\mathcal{G}^+$) and negative ($\mathcal{G}^-$) feedback graphs are vital for capturing users' preferences (likes/dislikes on items and attribute values) expressed in the conversation.

(2) ***Both the policy network and the Q-network are critical to conversational planning.*** We design two variants: replacing the policy network with random action type selection (w/o Pol. net.); replacing the Q-network with entropy-based action selection (w/o Q-net.). Performance drops in both variants suggest both networks are crucial for hierarchical action selection, and the absence of an informed decision maker, either at the action type or the action level, leads to poor conversational planning.

(3) ***Guidance from*** `S`***-planner is crucial for*** `S`***-agent to achieve strategic conversational planning.*** Removing `S`-planner and training `S`-agent on sampled on-policy trajectories as in Deng et al. [4] degrades the performance, because sampled trajectories may bring cumulative errors and biased estimations [14, 15], resulting in suboptimal conversational planning. By contrast, the high-rewarded conversation plans from `S`-planner offers robust guidance for `S`-agent and boosts its capability for strategic planning.

K.10 Hyper-parameter Sensitivity

Exploration and Exploitation The exploration factor w controls the balance between exploration and exploitation. To study its impact, we set w from 0.0 (exploitation only) to 5.0 (mostly favours exploration) and plot the success rate in Figure 3. We find that the performances are generally worse with only exploitation, but too much exploration does not negatively affect the performance. This is probably because our conversation tree has a very small search space (ask and rec), so high exploration does not incur much cost and also ensures thorough evaluation of actions, while high exploitation may prevent the tree from discovering the optimal action and lead to myopic solutions.

Influence of MCTS rollouts To study the influence of MCTS rollouts, we set N from 1 (equivalent to disabling MCTS, as there is no selection and reward back-propagation when $N = 1$) from 50 and plot the success rate and training time (on a single Tesla V100 GPU) in Figure 4. Unsurprisingly, we find that more rollouts increases the chance of discovering the optimal trajectory and lead to better performance, while disabling MCTS shows the worst performance. Nevertheless, we should also note that more rollouts bring additional computational cost, and setting $N = 20$ can achieve a good trade-off between efficiency and performance.

K.11 Additional Analysis on Action Strategies

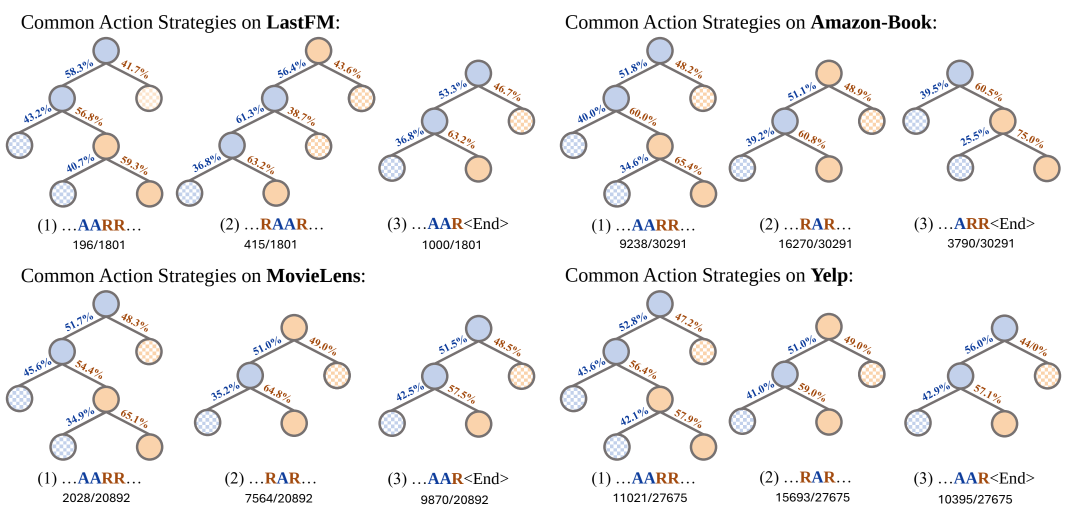

Figure 5: Common action strategies identified on four datasets. A stands for ask while R stands for recommendation. The probability of ask or recommendation at each node and the frequency of each action pattern (# of action patterns/# of users in this dataset) are shown in the figure.

To gain an insight into the planning and decision making ability of `SAPIENT`, we identify some typical action strategies of `SAPIENT` in Figure 5 that will be helpful for information seeking and recommendation success in the conversation.

- **...AARR...**: This action strategy occurs frequently during the conversation. The CRS first asks the user two questions consecutively to gather crucial information on user preference, and then quickly narrows down the candidate item list by making two targeted recommendation attempts.

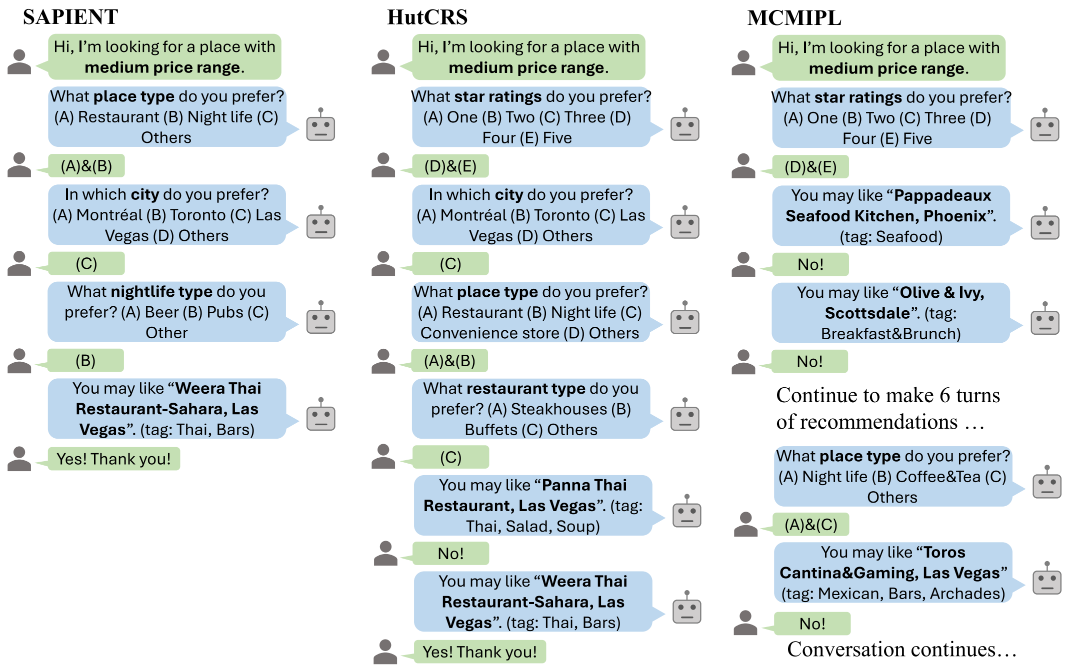

Figure 6: A case study on the Yelp dataset.

This strategy is useful as it not only addresses immediate user needs, but also incorporate long-term planning strategy to ensure that recommendations are personalized and relevant.

- **...RAAR...** and **...RAR...**: These are also two frequent action strategies during the conversation. In cases where an initial recommendation attempt fails, the CRS will adeptly adjust the action strategy by asking one or two additional questions to better understand the user's preference, ensuring that subsequent recommendations are more aligned with the user's needs. Interestingly, we find that on the LastFM dataset, the CRS tends to ask two additional questions, while on the other datasets, the CRS typically asks only one additional question. This is probably because the LastFM dataset has a very large number of attribute values, so two additional questions are required to fully clarify the user's preference.
- **...ARR<end>** and **...AAR<end>**: These two strategies occur frequently at the end of the conversation. Once the CRS has gathered sufficient information about the user's preferences, it is able to reach successful recommendation with only one or two attempts. This strategy enables the CRS to swiftly hit the target item, thereby shortening the conversation and reducing repeated recommendations.

K.12 Case Study

We provide a case study from a randomly sampled user in Figure 6 to demonstrate how `SAPIENT` makes strategic conversation planning. The user, who has visited some Thai restaurants before, is now looking for a nightlife venue in this conversation. `SAPIENT` quickly grasps user preference by asking only three questions, and makes a successful recommendation on the first attempt. By comparison, HutCRS can also make successful recommendations but requires more questions and recommendation attempts, while MCMIPL repeatedly makes failed recommendations. Owing to the collaborative graph encoder in the GPM, `SAPIENT` can infer user preferences from historical visits (e.g, the user's preference on Thai food) without the need for explicitly queries, thus reducing conversation turns and improving the comprehension of user preferences. Moreover, the progression from broad questions (e.g., place type) to specific questions (e.g., nightlife type) exemplifies how `SAPIENT` strategically plans conversations and asks information-seeking questions, with the policy network focusing on conversation action planning and the Q-network specializing in precise attribute value estimation. This design helps to narrow down candidate items quickly and ultimately improve the recommendation success rate.

L Limitations

First, our framework only supports searching over two types of actions (`ask` and `rec`) so far, which cannot search at a more fine-grained level (e.g., defining action types as "recommending items with a five-star rating", "recommending items with a three-star rating", rather than just "recommending items"). For future work, we plan to adopt advanced action abstraction techniques [47] to divide the search space at more fine-grained levels. Second, training `SAPIENT` requires conducting multiple simulated rollouts for each user, which introduces additional computational cost compared with existing methods. For future work, we plan to further improve the training efficiency of `SAPIENT` by adopting parallel acceleration methods [46] for MCTS. Third, while the user simulator approach can provide high quality simulations for the conversation [9, 48, 11], it may not fully represent the dynamics and complexities of user behaviors in the real-world situations. This issue may be partially addressed by developing an LLM-based user simulator that fully utilizes the human-likeliness of LLMs to better simulate user behaviors in the conversation for future work.

While it is our aspiration that CRS can provide personalized and user-friendly recommendations if correctly deployed, we also acknowledge that unintended uses of CRS may pose concerns on fairness and bias issues [49], which may be a potential risk for CRS but can be mitigated with debiasing algorithms as in the literature [50, 51].

M Ethics Statement

All datasets used in this research are from public benchmark open-access datasets, which are anonymized and do not pose privacy concerns.