ChatSOP: An SOP-Guided MCTS Planning Framework for Controllable LLM Dialogue Agents

Zhigen Li[1,2], Jianxiang Peng[1], Yanmeng Wang[2], Yong Cao[3], Tianhao Shen[1]
Minghui Zhang[1], Linxi Su[1], Shang Wu[4], Yihang Wu[1], Yuqian Wang[1]
Ye Wang[2], Wei Hu[2], Jianfeng Li[2], Shaojun Wang[2], Jing Xiao[2]
Deyi Xiong[1]*
[1]TJUNLP Lab, College of Intelligence and Computing, Tianjin University
[2]Ping An Technology [3]Tübingen AI Center, University of Tübingen
[4]Kunming University of Science and Technology
{zhigenli, dyxiong}@tju.edu.cn

Abstract

Dialogue agents powered by Large Language Models (LLMs) show superior performance in various tasks. Despite the better user understanding and human-like responses, their **lack of controllability** remains a key challenge, often leading to unfocused conversations or task failure. To address this, we introduce Standard Operating Procedure (SOP) to regulate dialogue flow. Specifically, we propose **ChatSOP**, a novel SOP-guided Monte Carlo Tree Search (MCTS) planning framework designed to enhance the controllability of LLM-driven dialogue agents. To enable this, we curate a dataset comprising SOP-annotated multi-scenario dialogues, generated using a semi-automated role-playing system with GPT-4o and validated through strict manual quality control. Additionally, we propose a novel method that integrates Chain of Thought reasoning with supervised fine-tuning for SOP prediction and utilizes SOP-guided Monte Carlo Tree Search for optimal action planning during dialogues. Experimental results demonstrate the effectiveness of our method, such as achieving a 27.95% improvement in action accuracy compared to baseline models based on GPT-3.5 and also showing notable gains for open-source models. Dataset and codes are publicly available.[1]

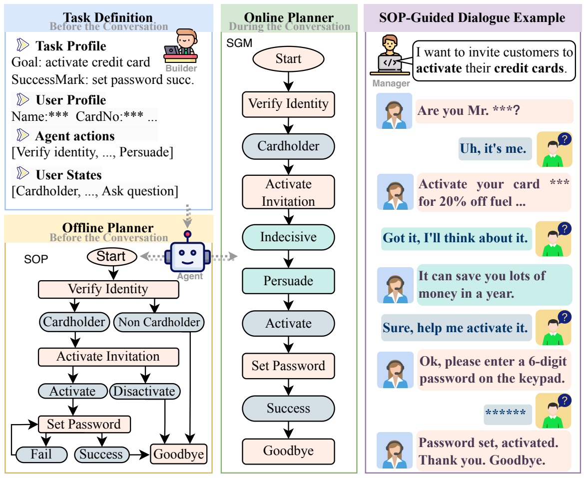


Figure 1: The three-stage workflow of our SOP-based planning framework for task-oriented dialogue with LLMs includes: (1) task definition, (2) the offline planner predicts the standard operating procedure (SOP) before the conversation, and (3) the online planner selects a dialogue action via SOP-guided MCTS (SGM) during the conversation. Control-sensitive actions are highlighted in orange. The right panel illustrates a dialogue in which the agent successfully activates a bank credit card by following the predicted SOP.

1 Introduction

Task-oriented dialogue agents are essential for applications such as hotel booking, technical support, and customer service (Ouyang et al., 2022; Moradshahi et al., 2023b). Recent advancements leverage Large Language Models' (LLMs) in-context learning ability to improve understanding, generate human-like responses, and adapt to diverse domains (Liu et al., 2024; Yi et al., 2024).

However, despite the enhanced intelligence powered by LLMs, a key challenge that persists in current dialogue agents is the lack of controllability (Achiam et al., 2023; Hadi et al., 2024). For instance, as illustrated in Figure 1 right panel, activating a credit card requires a specific sequence of steps—verifying personal information, creating a password, and activating the account. Omitting any step could result in task failure. Therefore, developing effective mechanisms to ensure greater control and goal-directed actions is crucial for LLM-based task-oriented dialogue agents.

To address this challenge, we introduce a Standard Operating Procedure (SOP) to regulate the dialogue flow strictly following the task process. Specifically, we propose **ChatSOP**, an SOP-guided Monte Carlo Tree Search planning framework designed to **enhance the controllability of LLM-**

*Corresponding author
[1]https://github.com/tjunlp-lab/ChatSOP

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 17637–17659

driven dialogue agents. Unlike methods relying on manually annotated dialogue flows or training data, our approach requires only user-provided task definitions and goals, enabling autonomous planning for better generalizability at low cost.

To define procedures required by specific tasks, we first introduce a standard operating procedure (SOP) to control the dialogue states inspired by the traditional dialogue framework (Anantha et al., 2021; Zhu et al., 2022), and construct a dataset designed for multi-scenario conversations, consisting of task descriptions, controlled SOP, and complete dialogues. This dataset is constructed through a four-step role-playing system utilizing GPT-4o, combined with human validation and modification to ensure intermediate dialogue control and data quality. To the best of our knowledge, this is the first dataset that provides SOP intermediate annotations, which could also be explored to evaluate general-purpose LLM agents.

To enable LLM dialogue agents to complete goal-driven tasks with controllability, we propose a three-stage SOP-based planning framework. As illustrated in Figure 1, when provided with the task definition and dialogue goal, an offline planner predicts a task-specific Standard Operating Procedure before the conversation. During the conversation, an online planner leverages SOP-guided Monte Carlo Tree Search (SGM) to select the optimal action that not only follows the SOP but also proactively guides the user toward the dialogue goal.

Experimental results demonstrate that our method achieves significant improvements in task success rate, with a 27.95% increase in overall action accuracy compared to baseline based on GPT-3.5. Additionally, for open-source models, larger models yield substantially better results, as evidenced by the performance gap between Llama3-70B (78.35%) and Llama3-8B (46.85%), highlighting their ability differences in dialogue tasks.

In summary, we make **three main contributions**: 1) We develop a semi-automatic role-playing framework with manual review, then construct the first SOP-annotated dataset to support research on controllable dialogue agents. 2) We propose a planning-based framework integrating SOP and MCTS to enhance controllability of LLM task-oriented dialogue; 3) Extensive experiments via automatic and human evaluation demonstrate the utility of our dataset and effectiveness of our method, achieving superior performance in offline SOP prediction and online dialogue planning.

2 Related Work

Dialogue Agents. Existing approaches to dialogue agents can be categorized into four groups: conversational question answering (CQA) (Singhal et al., 2023; Shi et al., 2024; Liu et al., 2023), open-domain dialogue (ODD) (Ouyang et al., 2022; Zhang et al., 2023; Liu et al., 2023), task-oriented dialogue (TOD) (Quan et al., 2020; You and Xiong, 2024), and conversational recommender systems (CRS) (Zhang et al., 2021; Wang et al., 2023a). CQA and ODD passively respond to users with knowledgeable or engaging conversations. TOD provides functional services following a structured process driven by training data (Budzianowski et al., 2018; Quan et al., 2020; Moradshahi et al., 2023a). CRS plans dialog actions to guide conversations toward given goals (Wu et al., 2019) but often fails to handle complex tasks requiring strict sequential constraints (Akyar, 2012; Zhou et al., 2023).

Dialogue Planning and Policy Optimization. Traditional dialogue planning research has focused on subgoal generation (Zhang et al., 2021), the next round of dialogue transition strategy (Tang et al., 2019), hierarchical strategy (Kishinami et al., 2022). While Recent frameworks explore planning dialogue paths using basic knowledge, goal-oriented dialogue planning frameworks, and proactive transitions between dialogue stages (Wang et al., 2022a). Reinforcement Learning (RL) has long been a cornerstone for optimizing dialogue policies. The advent of LLMs has significantly advanced this area, enabling the use of step-by-step RL for task-oriented dialogue (Du et al., 2024), the development of more proactive systems (Dong et al., 2025), and the introduction of a dual-process planner framework combining LLMs and MCTS for policy optimization (He et al., 2024). However, a persistent challenge, as highlighted by Wang et al. (2023b), is that many approaches employ greedy single-turn prediction strategies but ignore the interdependencies of global policies, resulting in uncontrollability from the perspective of global conversation. Thus, we propose a SOP-guided planning approach to address this issue.

Planning and Reasoning of LLMs. LLMs show prowess in planning and reasoning. Examples include Chain-of-Thought (Kojima et al., 2022a), its variants (Kojima et al., 2022b) , Self-Consistency (Wang et al., 2022b), Least-to-most Prompting (Zhou et al., 2022) and Self-Assessment

(Welleck et al., 2022; Shinn et al., 2023). Recent efforts have used more complex reasoning processes, offering new avenues to improve and optimize the reasoning process (Zhang and Xiong, 2025). For example, Yao et al. (2023) apply heuristic-based search methods, such as depth-first and breadth-first search, to discover optimized reasoning pathways. Zhu et al. (2022) and Hao et al. (2023b) have introduced MCTS to reason steps for complex math or logical reasoning. Unlike them, we use MCTS for dialogue planning, encoding SOP constraints into its expansion and simulation steps.

3 Problem Formulation

In our work, we decompose the dialogue tasks into three steps: task initialization, Standard Operating Procedures (SOPs) prediction, and task execution via dialogue generation based on SOPs.

Task Initialization. When a user specifies a task, we collect the user-defined task profile and associated user information profile, denoted as p. These profiles include textual descriptions of the goal of the task, relevant background knowledge, and user-specific information. We define a multi-turn dialogue as $D_t = \{(u_t, s_t, a_t, r_t)\}_{t=1}^{T}$, where each tuple (u_t, s_t, a_t, r_t) denotes the t-th turn of the dialogue. Here, u_t is a user utterance, s_t represents user states, a_t denotes agent actions, and r_t is the agent response utterance. Please refer to Appendix A.1 for detailed examples.

Then, as shown in Figure 1(a), we define the SOP graph G_t as a directed graph, where the vertices are annotated with agent actions a_t and user states s_t, and the edges represent the connections between these vertices.

SOP Prediction. As an intermediate step for controlling dialogue generation, SOP prediction is to predict the connections of SOP graph nodes with given user states s_t and agent actions a_t, enabling the construction of a complete SOP graph G. To do so, we introduce the adjacency List $\mathcal{M}$ to represent all the connections in the SOP graph. Thus, the task is defined as follows:

$$\hat{m} = \arg\max_{m \in \mathcal{M}} P(m \mid s_t, a_t), \quad (1)$$

It is important to emphasize that any modifications or deletions to the SOP graph will result in inaccurate task completion.

Dialogue Generation. The aim of this task is to first generate user states at turn $t+1$ based on profiles p, historical dialogue D_t with the predicted SOP graph G_t and user utterance u_{t+1}:

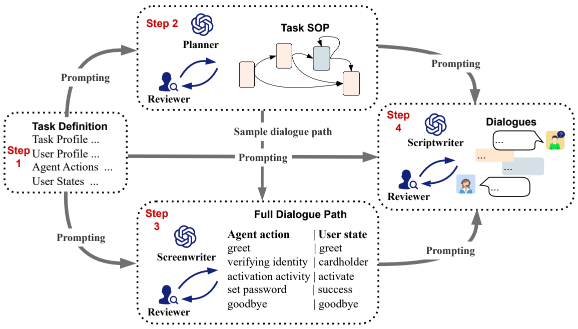


Figure 2: Overview of our role-playing framework for *SOPDAIL* dataset construction. This process uses human-LLM collaboration across four steps: (1) An annotator provides the initial task definition. (2) An LLM acts as a "Planner" to generate a candidate SOP graph, which is then reviewed by the human. (3) An LLM "Screenwriter" samples a path from the SOP and enriches it. (4) An LLM "Scriptwriter" generates the final dialogue script, which is again validated by a human reviewer. All LLMs are GPT-4o.

$$\hat{s}_{t+1} = \arg\max_{s_{t+1}} P(s_{t+1} \mid D_t, p, u_{t+1}, \mathcal{M}) \quad (2)$$

Then, we predict the agent action and generate a response at turn $t+1$ as follows:

$$\hat{a}_{t+1} = \arg\max_{a_{t+1}} P(a_{t+1} \mid D_t, p, \hat{s}_{t+1}, \mathcal{M}) \quad (3)$$

$$\hat{r}_{t+1} = \arg\max_{r_{t+1}} P(r_{t+1} \mid D_t, p, u_{t+1}, \hat{a}_{t+1}, \mathcal{M}) \quad (4)$$

4 The ChatSOP Dataset

Before introducing our method, we present the creation of an SOP-annotated dialogue dataset, namely *SOPDAIL*, which includes designing a role-playing framework to simulate diverse scenarios, conducting human annotations to ensure high-quality, and performing a comprehensive dataset analysis.

4.1 Dataset Curation Framework

Inspired by recent studies (Wang et al., 2023a; Sandler et al., 2024) that directly use LLMs to create high-quality dialogue datasets, we introduce a role-playing framework where LLMs simulate various agent roles to generate task-oriented dialogues. As shown in Figure 2, our framework follows a four-step curation process: task definition, SOP planning, dialogue path creation, and dialogue generation. The detailed steps are as follows.

Dataset	Participants	Agent Goals	PA	CT	Lang	#Domains	#Tasks	#Dialogues
DSTC2 (Henderson et al., 2014)	Crowd	N/A	✗	✓	English	1	1	1,612
DSTC4 (Kim et al., 2017)	Experts	N/A	✗	✓	English	1	1	35
CrossWOZ (Zhu et al., 2020)	Rules,Crowd	N/A	✗	✓	Chinese	5	5	5,012
SGD (Rastogi et al., 2020)	Rules,Crowd	N/A	✗	✓	English	16	26	16,142
OTTers (Sevegnani et al., 2021)	Crowd	Topics	✓	✗	English	Open	1	4,316
TOPDIAL (Wang et al., 2023a)	LLM	Act-topic pairs	✓	✗	English	3	1	9,939
SOPDAIL (ours)	LLM, Experts	Open Definition	✓	✓	Chinese	32	53	3,114

Table 1: A comparison between our proposed *SOPDAIL* and other relevant datasets, where PA indicates whether it includes proactive interaction, CT denotes controllability, and Lang denotes language.

Step 1: To satisfy the diversity of our dataset, we curate 53 unique tasks, including activities such as activating a bank card, scheduling appointments, and online shopping, across 32 domains, e.g., shopping, education, hospital, etc. Please refer to Appendix Table 7 for all tasks and domains.

Step 2: To generate intermediate SOP annotations for SOP prediction, as defined in §3, we utilize zero-shot prompting to instruct LLMs in acting as **planners** to draft SOPs for the specified tasks.

Step 3: We then prompt LLMs to generate multi-turn dialogue paths sampled from the annotated SOPs. To ensure the dialogues reflect proactive interactions and closely mimic real-world scenarios, we assign LLMs the role of **screenwriters**, instructing them to insert predefined proactive agent actions (e.g., offering help, persuading) and user states (e.g., asking question) into dialogue paths.

Step 4: We assign LLMs as **scriptwriters** to draft dialogues for each agent action and user state, used to evaluate dialogue generation in §3.

Note: Human annotators are involved in reviewing and refining the LLM-generated annotations in Steps 2-4, ensuring their accuracy and quality. Besides, a user simulator generates a unique user profile and updates the *task definition* in step 3 and 4, enabling diverse dialogue paths and interactions. Detailed prompts are provided in Appendix A.3.1.

4.2 Human Annotation

Annotator Selection. We recruited seven annotators with relevant qualifications and expertise to ensure the quality of the annotation process. Before annotation, all participants are trained to gain a thorough understanding of the annotation guidelines. The annotators worked independently but were allowed to provide feedback or reject any doubtful cases. On average, the annotation time was 10.3 minutes per sample, and annotators were compensated at a rate of $8 per hour.

Annotation Process. We develop an in-house web application as the annotation platform. Participants are required to read the guidelines, pass the pre-annotations, and then perform the actual annotations. Additionally, every instance was assigned to three annotators for cross-annotation validation with an inter-annotator agreement (IAA) score of 0.88, showing high consistency of annotation. Instances with an IAA below 0.95 were excluded. Please refer to Appendix A.3.2 for more details.

4.3 Dataset Analysis

Comparison with Existing Datasets. Table 1 provides a comparison of *SOPDAIL* against other relevant datasets, highlighting the distinct advantages of our dataset. Notably, *SOPDAIL* covers 53 tasks across 32 domains, offering a comprehensive and diverse evaluation framework for dialogue agents. Furthermore, this dataset is well-suited for in-context learning with LLMs (Yu et al., 2023), as opposed to traditional fine-tuning approaches (Kojima et al., 2022b), thus enhancing the efficiency of task deployment. To the best of our knowledge, *SOPDAIL* is the first Chinese benchmark for proactive and controllable dialogue. We anticipate this dataset will serve as a valuable resource for advancing research on controllable LLM-driven agents.

Statistics. Table 2 summarizes the statistics of our *SOPDAIL*. The quantity analysis shows that 74% of the utterances are for controllability, aligning well with the objective of our work. To assess the quality of the dataset, we measured the accuracy of samples after annotation by human experts. Three annotators were invited to evaluate 300 randomly sampled cases, rating each as 1 if it adhered to the instructions and was semantically correct, and 0 otherwise. The results indicate a 0.98 accuracy score, underscoring the high quality of our dataset. Notably, even the raw results achieved a 0.91 accuracy rate, highlighting the superior per-

Quantity Statistics	
Total # Domains / Tasks / Goals	32 / 53 / 70
Total # SOP vertices / Edges	899 / 1,058
Total # Dialogues / Turns	3,114 / 23,897
Total # Utterances / Tokens	47,795 / 119,5736
Avg. # Turns / Utterances per dialogue	7.67 / 15.34
Avg. # Words per utterances	25.01
Rate. # Controllability / Proactivity	0.74 / 0.26
Quality Statistics	
Before Expert Correction	
Avg. # *SOPDAIL* Sample Accuracy	0.91
After Expert Correction	
Avg. # ED of dialogue paths / utterances	0.41 / 0.34
Avg. # *SOPDAIL* Sample Accuracy	0.98

Table 2: The statistics of our *SOPDAIL*, where ED represents the edit distance used to evaluate the discrepancy between raw results and those after expert corrections. The high accuracy demonstrates the superior performance of LLMs and the high quality of our dataset.

formance of GPT-4o in this task.

5 Our Approach

In this section, we present the details of our proposed method, beginning with a multi-turn dialogue framework powered by LLMs. We then discuss the fine-tuning method for SOP prediction, followed by leveraging Monte Carlo Tree Search (MCTS) for dialogue generation.

5.1 Framework Overview

As illustrated in the left panel of Figure 3, the planning-based dialogue agent (ChatSOP) consists of five components: 1) *LLMs Module*: Responsible for managing and utilizing multiple LLMs to support various functionalities; 2) *Dialogue State*: Handles the storage and update of task prompts and dialogue history. 3) *SOPs Pool*: Contains predefined SOP vertices and predicted edges; 4) *Offline Planner*: Constructs an adjacency list to assemble a complete SOP graph from the provided task definition; 5) *Online Planner*: Generates the dialogue based on the predicted SOP graph.

Given a *task definition* from the user, the agent initially retrieves relevant SOP nodes from the *SOPs Pool* through an iterative search. The *Offline Planner* is then employed to generate an adjacency list representing a complete SOP graph. Once the graph is constructed, the *Dialogue State* module is activated to prepare task-specific prompts and manage dialogue history. Finally, the *Online Planner* generates the dialogue using the SOP graph and the prepared prompts. It is important to note that both the Offline and Online Planners are powered by the *LLMs Module*. Below, we present the implementation of the offline and online modules.

5.2 Offline Planner

The objective of this module is to predict the adjacency List $\mathcal{M}$ based on user states and agent actions, thereby guiding user interactions with the agent. To achieve this, we propose three methods: Direct Adjacency List (DAL), Translation Chain-of-Thought (TCoT), and Supervised Fine-Tuning (SFT). Specifically, DAL employs direct prompting of LLMs to generate an adjacency list in JSON format. Here, an adjacency list represents the connectivity between two vertices, where a value of 1 indicates a connection and 0 indicates no connection. TCoT involves a two-step process: first, the LLMs are prompted to describe each vertex and its child vertices in natural language, including justifications for the relationships; then, the description is translated into an adjacency list in JSON format. Finally, SFT implements fine-tuning on various LLMs, such as Llama and Qwen, to iteratively generate the adjacency vertices for each vertex in the SOP graph. Please refer to Appendix A.7 for detailed prompt settings.

5.3 Online Planner

Following the prediction of the SOP graph, we now delve into dialogue generation driven by the online planner module. Specifically, the target is to predict the practical dialogue path, as shown in the right panel of Figure 3, where nodes represent dialogue states d_t, including both agent actions a_t and user states s_t. At each step, the agent is required to predict the next actions based on its working memory. However, direct use of exhaustive search over the entire space can lead to sub-optimal dialogue paths. Thus, in our work, we propose **S**OP-**g**uided **M**onte Carlo Tree Search (**SGM**) to construct the dialogue path, assuring to predict the optimal action through N steps simulations.

Given the initial dialogue state d_0, inspired by Hao et al. (2023a), we propose an iterative process to search for the optimal next action in 4 steps: node **selection**, node **expansion**, dialogue **simulation**, and **back-propagation**. After n iterations, the optimal next action for d_0 is selected. This process continues until the predefined computational budget is reached (e.g., number of iterations), at which point the resultant trajectory can be extracted from

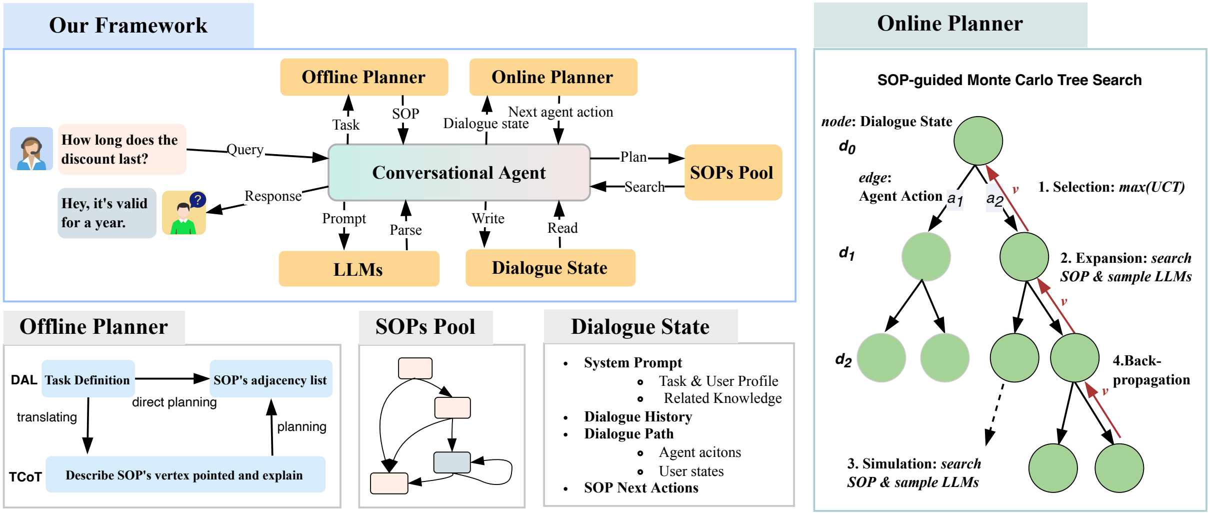


Figure 3: Diagram of the ChatSOP framework. The system has five components: (1) **LLMs** Module for language models management; (2) **Dialogue State** for history management; (3) **SOPs Pool** for SOP storage; (4) the **Offline Planner**, which constructs the SOP graph before a conversation; and (5) the **Online Planner**, which generates dialogue in real-time using the SOP. **Workflow:** Before a conversation, the Offline Planner builds the SOP and stores it. During the conversation, the Online Planner retrieves the SOP to guide its MCTS-based response generation.

the tree. The algorithm is provided in Appendix 1. Below, we provide the details of each stage.

Selection. Starting from the root node (i.e., the initial state d_0), a child node is selected at each level of the tree to determine the next state. This process continues until it reaches a leaf node. To balance exploration (less-visited nodes) and exploitation (high-value nodes), we employ the Upper Confidence Bounds for Trees (UCT) algorithm (Kocsis and Szepesvári, 2006) for child node selection:

$$a_t^* = \arg\max_{a_t} \left[Q(d_t, a_t) + w \sqrt{\frac{\ln N(d_t)}{N(c(d_t, a_t))}} \right] \quad (5)$$

where $N(d_t)$ denotes the number of times node d_t has been visited in previous iterations, $c(d_t, a_t)$ is the child node of applying a_t in state d_t and w denotes an empirical weight parameter (set to 1 in our experiments) to balance exploration and exploitation. The less a child node has been visited (i.e., the more uncertain the child node is), the higher the second term.

The state-action value function $Q(d_t, a_t)$ estimates the expected future reward associated with taking action a_t in state d_t:

$$Q(d_t, a_t) = \lambda \cdot (\mathcal{L}(d_t, a_t) + (1-\lambda) \cdot \mathrm{T}(d_t, a_t)) \quad (6)$$

where $\mathcal{L}(d_t, a_t)(0 \leq \mathcal{L}(d_t, a_t) \leq 1)$ function represents the logical rationality of the current action, as assessed by the LLMs. It is computed as the mean of several binary (0 or 1) evaluations derived from prompt-based sampling by the LLMs. The $\mathrm{T}(d_t, a_t)$ function assigns discrete values to measure task completion: 0.3 for the termination state, 0.7 for the success state, and 0 for others. λ is a hyperparameter, set to 1 in our experiments, that balances logical rationality and task completion.

Expansion. After a leaf node (non-terminate) is selected, the agent samples m possible dialogue states d_t iteratively for expansion. Notably, we first utilize the local subgraph from SOP graph, then add the next two levels of child nodes connected to the current state node for further expansion. This setting ensures that the agent maintains a balance between constraints and proactivity. Finally, when the selected leaf node is already a terminal node (either a dialogue end node or the maximum search depth has been reached), we will skip the expansion phase and proceed to back-propagation.

Simulation. To estimate the reward generated by future dialogue, we simulate the future dialogue for each expanded state node. To improve efficiency, we follow a process similar to the expansion phase mentioned above, that is, we only simulate downward for candidate dialogue policies that are sampled from LLM and guided by the SOP.

Backpropagation. At the final step, once a terminal state is reached, the Q values are updated along the entire dialogue path. The algorithm terminates when the predetermined total number of iterations is completed. Finally, within the con-

Task	Train	Valid	Test
SOP Prediction	31	5	17
Dialogue Generation	1,859	324	931

Table 3: *SOPDAIL* dataset statistics for training, validation, and test splits across different domain tasks. SOP Prediction utilizing 5-fold cross-validation.

structed dialogue tree, the child node with the highest Q value of the current node is selected to guide the next turn in the conversation. The details are provided in Algorithm 1 in Appendix A.8.

6 Experiments

We conducted extensive experiments to validate our curated dataset and planning-based dialogue agent with both automatic and human evaluations.

6.1 Datasets

We split *SOPDAIL* dataset into training, validation, and test sets, as shown in Table 3. To evaluate the methods on unseen tasks, we split the data at the task level rather than the dialogue level, thereby avoiding any task overlap between the three sets. Additionally, we employed 5-fold cross-validation for SOP prediction to ensure result validity.

6.2 Experimental Setting

Baseline Setting For SOP prediction, we compare our method (TCoT and SFT) against the baselines DAL. For dialogue generation, we compare our method (SGM) with CoT and CoT+SOP. The LLMs include GPT-3.5-turbo/4o, Qwen1.5-14b/72b-chat, and Llama3-8b/70b-chat, covering both open and closed models across different sizes.

Paramerters We use the GPT models[2] through the provided API, while for open-source models, we directly load the pre-trained versions from HuggingFace models[3], experiments on 4 Nvidia A800 GPUs. The inference is performed with a temperature and top-p setting of 0.1. For SFT, we fine-tuned all parameters using 5 epochs, 50 warm steps, 1e-6 learning rate, and 128 batch size. Hyperparameters are in Appendix A.4

Evaluation Metrics We evaluated SOP prediction in terms of graph structure and dialogue usability. For graph structure, we calculated the graph edit distance (GED) and its operation ratio (GEDR) (Hagberg et al., 2008) needed to match the predicted SOP with the ground truth. For dialogue usability, we measured the precision, recall, and F1 score of the dialogue paths, defining true positives as paths in the predicted SOP that exactly match the ground-truth paths. For dialogue generation, we measured the accuracy of single-turn dialogue actions (Acc T), controllable actions (Acc C), and proactive actions (Acc P) to evaluate the dialogue's controllability and proactivity. The accuracy of the entire dialogue (Acc D) was calculated by dividing the number of correct dialogues (i.e., each turn is correct) by the total number of dialogues.

[2]GPT: https://platform.openai.com/docs/models
[3]HuggingFace: https://huggingface.co/models

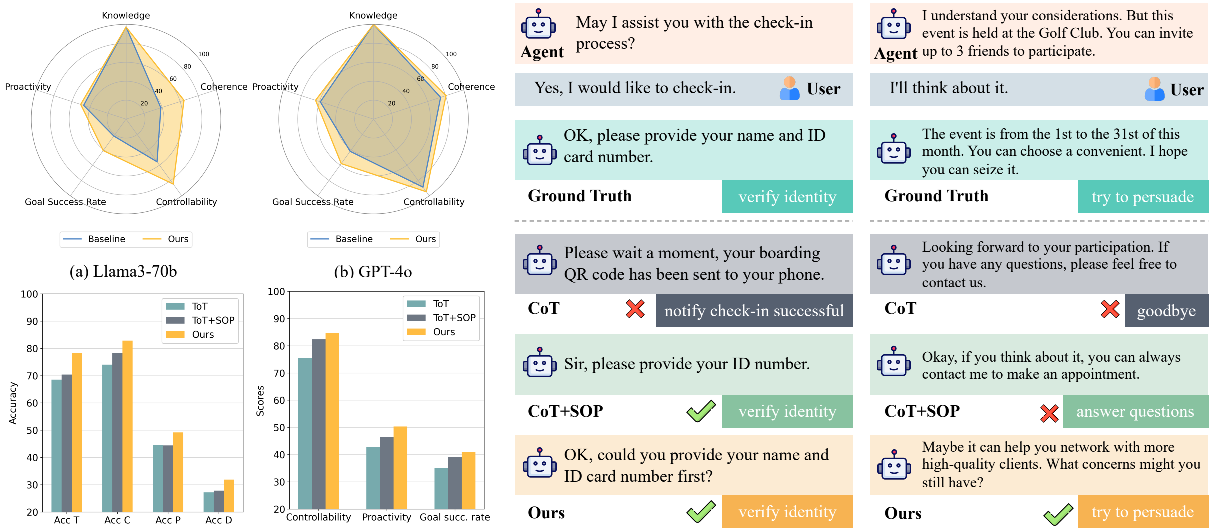
Model	Method	Pre↑	Rec↑	F1↑	GED↓	GEDR↓
GPT-4o	DAL	**78.11**	66.51	**71.85**	**2.01**	**5.40**
	TCoT	69.20	**73.34**	71.22	4.70	12.46
GPT-3.5	DAL	41.39	21.94	28.68	7.61	22.48
	TCoT	**50.95**	**50.03**	**50.48**	**6.91**	**18.28**
Qwen1.5-14b	DAL	42.84	37.81	40.17	10.23	25.77
	TCoT	38.79	42.69	40.65	11.36	26.91
	SFT	**68.58**	**62.73**	**65.52**	**3.94**	**10.30**
Qwen1.5-72b	DAL	46.74	31.57	37.68	10.75	28.32
	TCoT	48.29	51.94	50.04	7.38	18.72
	SFT	**80.25**	**74.01**	**77.00**	**2.86**	**7.12**
Llama3-8b	DAL	35.19	28.04	31.21	8.31	24.05
	TCoT	44.50	40.35	42.32	10.92	30.10
	SFT	**72.19**	**68.33**	**70.21**	**3.54**	**9.33**
Llama3-70b	DAL	64.14	56.05	59.82	3.85	10.72
	TCoT	60.81	67.34	63.91	5.43	14.38
	SFT	**74.10**	**73.03**	**73.56**	**2.81**	**7.27**

Table 4: Results for SOP prediction, where Pre and Rec are precision and recall, while GED and GEDR are the graph edit distance and its editing ratio, showing that SFT significantly outperforms the baselines.

6.3 Main Results

SOP Prediction Table 4 presents the results of our methods (TCoT and SFT) compared to DAL. The TCoT outperforms the baseline DAL on most metrics, particularly showing significant improvements in recall and F1 scores. This indicates that TCoT's predicted SOP has a stronger advantage in dialogue usability. Similarly, our SFT method also shows significant improvements over all metrics and models. Notably, the fine-tuned Qwen1.5-72b achieves the highest score, even surpassing GPT-4o (*77.00* vs *71.85* in F1 score), underscoring the effectiveness of our proposed dataset.

Dialogue Generation For dialogue action generation, as shown in Table 5, our method (SGM) substantially outperforms the baseline methods across all metrics and models. GPT-3.5 based agent achieves a score of *76.52* compared to *48.57* by



(a) Llama3-70b (b) GPT-4o (c) Automatic Evaluation (d) Human Evaluation (e) Case 1: CoT+ SOP and Ours succeed (f) Case 2: Only Ours succeed

Figure 4: More analysis results: (a)-(b) presenting human evaluation results for Llama3-70b and GPT-4o models based on the five proposed metrics; (c)-(d) comparing ToT with our method (SGM) through both automatic and human evaluations, demonstrating the superiority of our approach; and (e)-(f) providing two case studies to further illustrate the advantages of our method.

Model	Method	Acc T ↑	Acc C ↑	Acc P ↑	Acc D ↑
GPT-4o	CoT	69.64	74.34	51.30	22.72
	CoT+SOP	82.09	86.73	57.14	43.39
	SGM (our)	**86.37**	**91.19**	**60.42**	**46.29**
GPT-3.5	CoT	48.57	55.75	18.52	4.83
	CoT+SOP	63.24	69.49	22.22	28.57
	SGM (our)	**76.52**	**80.42**	**33.10**	**38.45**
Qwen1.5-14b	CoT	50.47	55.75	25.93	2.15
	CoT+SOP	55.64	61.57	25.67	2.69
	SGM (our)	**61.76**	**67.11**	**28.75**	**3.33**
Llama3-8b	CoT	31.79	36.73	11.11	0.32
	CoT+SOP	38.72	43.28	17.14	2.26
	SGM (our)	**46.85**	**56.30**	**22.31**	**3.76**
Qwen1.5-72b	CoT	68.57	73.89	46.30	12.57
	CoT+SOP	74.25	79.65	45.24	30.72
	SGM (our)	**77.83**	**83.54**	**47.32**	**29.32**
Llama3-70b	CoT	65.43	72.52	44.22	11.71
	CoT+SOP	52.24	54.42	40.48	12.24
	SGM (our)	**78.35**	**82.86**	**49.18**	**31.87**

Table 5: Automatic evaluation results of dialogue generation. T, C, P, and D denote turn, controllable, proactive, and dialogue, respectively.

the baseline. Additionally, larger models demonstrate better performance, with GPT-4o achieving the highest scores, particularly excelling in controllable action generation with a score of *91.19*. These findings highlight the effectiveness of our proposed method in tackling such a challenging multi-scenario dataset and generating dialogues.

6.4 Human Evaluations

To further assess the quality of generated utterances, we conducted a human evaluation to measure the proportion of accurate control actions, proactive actions, and knowledge accuracy in each single-turn on 100 dialogues sampled from compare models. Besides, from a broader dialogue perspective, we further proposed two additional metrics: goal success rate, defined as correct actions, correct knowledge and goal achieved, and logical coherence score, defined as logic correct and consistent to history, both scored on a scale from 0 to 1. Detailed definitions are listed in Appendix A.5. The evaluation was conducted by the same annotators previously described in §4.2.

As illustrated in Figure 4(a)-(b), we compare the performance of our method with the baseline across two models. For Llama3-70b, our method demonstrates significantly superior performance, with particularly notable improvements in controllability, goal success rate, and coherence. For GPT-4o, while the baseline model already achieves strong results, our method can still enhance scores across multiple dimensions, underscoring its effectiveness. Notably, in the knowledge dimension, all methods achieve consistently high scores, indicating that the models possess sufficient knowledge to support task-oriented dialogues effectively.

6.5 Analysis Experiments

Effect of Offline Planner To better understand how offline planners affect SGM, we selected SOPs with varying F1 scores from the results of different online planners as input for SGM's dialogue prediction. Figure 5 shows the correlation between

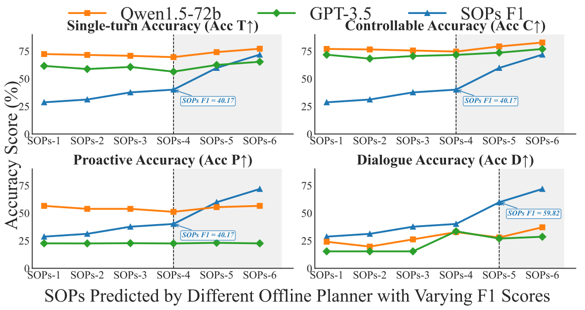


Figure 5: Impact of SOP's accuracy (F1) on dialogue evaluation metrics for our SGM with different LLMs. The gray area indicates that the SOP offline planner's improvement can enhance the performance of SGM.

four dialogue evaluation metrics and SOP-F1. The statistical results reveal strong positive Spearman correlations between SOP-F1 and overall dialogue accuracy (Acc D): 0.88 for Qwen and 0.75 for GPT. More specifically, it can be observed from see shaded area in Figure 5 that the impact of SOPs is negligible when their F1 scores are below 40.17 or 59.82, but progressively increases above these thresholds. This indicates that higher SOP accuracy corresponds to better dialogue performance.

Effect of Online Planner To evaluate the performance of different tree search algorithms for online planner, we implemented Tree-of-Thought (ToT) for comparison with Monte Carlo Tree Search (MCTS). Figures 4(c)-(d) show that our SGM yields superior results in automatic and human evaluations. Moreover, incorporating SOP with ToT enhances performance, emphasizing the consistent benefits of SOP integration.

Cost-Effectiveness Analysis We analyze the cost-effectiveness of different online planning algorithms with the same SOP offline planner, by measuring average *token*, *time*, and *monetary* costs per conversational turn on LLaMA3-70B-Chat. Compared to CoT, SGM improves goal success rate by 19% with a cost increase from \$0.055 to \$0.456 per turn (roughly 8×). Compared to ToT, SGM yields a slightly higher success rate (41% vs. 39%) while being notably more efficient—saving 8.64 seconds and \$0.224 per turn. These results suggest that SGM achieves a better trade-off between performance and efficiency.

Comparison with RL-based Dialogue Policy We conducted a comparative experiment with the state-of-the-art RL-based dialogue policy method PDPD (He et al., 2024), which also utilizes dual planners: an LM-based planner and an MCTS-based planner activated under uncertainty. For fair comparison, we adapted PDPD to an unsupervised setting consistent with our SGM framework, with aligned hyperparameters. The results from Table 6 show that our SGM method achieves consistently higher accuracy across all dialogue policy metrics, especially in controllable policy selection.

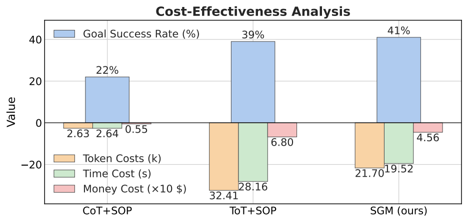


Figure 6: Comparison of cost-effectiveness among different online planners per conversational turn. *Money Costs* are scaled ×10 for better visibility.

Model	Method	Acc T ↑	Acc C ↑	Acc P ↑	Acc D ↑
GPT-4o	DPDP	72.29	72.81	56.51	24.50
	SGM (our)	**86.37**	**91.19**	**60.42**	**46.29**
Qwen1.5-72b	DPDP	69.92	72.79	47.27	16.05
	SGM (our)	**77.83**	**83.54**	**47.32**	**29.32**

Table 6: Automatic evaluation results of PDPD and our SGM. T, C, P, and D denote turn, controllable, proactive, and dialogue, respectively.

6.6 Case Study

Furthermore, Figures 4(e)-(f) provide case studies comparing the dialogue actions selected and responses generated by different methods in the same contexts. In the first case, verifying identity is a prerequisite before checking in. After applying SOP, both CoT and ours select actions aligned with SOP guidelines. However, in scenarios where the optimal action is absent from the SOP, such as proactive persuasion in the second case, CoT+SOP fails to continue the persuasion attempt. In contrast, our method can leverage simulation and deeper dialogue path exploration to select a more goal-oriented action, resulting in a more effective persuasion strategy.

7 Conclusion

In this paper, we have presented a planning-based framework, a high-quality benchmark dataset, and an unsupervised algorithm that encodes SOP constraints into Monte Carlo Tree Search for controllable LLM dialogue agents. Based on LLMs without additional training, our approach offers a better controllable and scalable solution for enterprise-level dialogue systems.

Limitations

Hallucinations Our approach is based on the context learning of LLMs, such as ChatGPT and GPT-4. As LLMs may produce outputs containing hallucinations (Bang et al., 2023), our system might provide information beyond the task definition. We intend to enhance the veracity of responses through post-processing steps, such as training a dedicated safety model and incorporating checks and revisions into the post-processing phase.

Runtime One significant limitation of our method is the runtime. The more exhaustive the tree search is (e.g., increasing n or k), the more likely the algorithm is to find the optimal dialogue policy. However, this comes at the cost of longer simulation times, which may impact the overall user experience. We believe that parallelizing the tree search or reusing portions of the simulated subtrees could help to speed up the runtime. We anticipate that with the advancement of LLMs research, the speed of inference will continue to improve.

Ethics Statement

Given the independent behavior of agents in goal-oriented dialogue, it's imperative to scrutinize ethical implications. Our approach does not force the agent to achieve a specified goal, nor does it force the user to accept the agent's request. Instead, our work highlights the criticality of directing agents to adhere to human-defined limitations. While our measures are potent, we advocate for the stringent regulation of goal signals, particularly when implementing goal-oriented dialogue systems in specialized fields. Currently, the targeting process must uphold factual accuracy, respect user privacy norms, and comply with societal laws.

Acknowledgments

The present research was supported by the National Key Research and Development Program of China (Grant No. 2024YFE0203000). We would like to thank all the anonymous reviewers for their constructive and insightful feedback, which has greatly helped to improve the quality of this paper.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. GPT-4 Technical Report. arXiv preprint arXiv:2303.08774.

Isin Akyar. 2012. Standard operating procedures (what are they good for ?). In Isin Akyar, editor, Latest Research into Quality Control, chapter 17. IntechOpen, Rijeka.

Raviteja Anantha, Svitlana Vakulenko, Zhucheng Tu, Shayne Longpre, Stephen Pulman, and Srinivas Chappidi. 2021. Open-domain question answering goes conversational via question rewriting. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 520–534, Online. Association for Computational Linguistics.

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, et al. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. ArXiv preprint, abs/2302.04023.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.

Wenjie Dong, Sirong Chen, and Yan Yang. 2025. ProTOD: Proactive task-oriented dialogue system based on large language model. In Proceedings of the 31st International Conference on Computational Linguistics, pages 9147–9164, Abu Dhabi, UAE. Association for Computational Linguistics.

Huifang Du, Shuqin Li, Minghao Wu, Xuejing Feng, Yuan-Fang Li, and Haofen Wang. 2024. Rewarding what matters: Step-by-step reinforcement learning for task-oriented dialogue. In Findings of the Association for Computational Linguistics: EMNLP 2024, pages 8030–8046, Miami, Florida, USA. Association for Computational Linguistics.

Muhammad Usman Hadi, Qasem Al Tashi, Abbas Shah, Rizwan Qureshi, Amgad Muneer, Muhammad Irfan, Anas Zafar, Muhammad Bilal Shaikh, Naveed Akhtar, Jia Wu, et al. 2024. Large language models: a comprehensive survey of its applications, challenges, limitations, and future prospects. Authorea Preprints.

Aric Hagberg, Pieter J Swart, and Daniel A Schult. 2008. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Laboratory (LANL), Los Alamos, NM (United States).

Shibo Hao, Yi Gu, Haodi Ma, Joshua Hong, Zhen Wang, Daisy Wang, and Zhiting Hu. 2023a. Reasoning with language model is planning with world model. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pages 8154–8173, Singapore. Association for Computational Linguistics.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023b. Reasoning with language model is planning with world model. ArXiv preprint, abs/2305.14992.

Tao He, Lizi Liao, Yixin Cao, Yuanxing Liu, Ming Liu, Zerui Chen, and Bing Qin. 2024. 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, Bangkok, Thailand. Association for Computational Linguistics.

Matthew Henderson, Blaise Thomson, and Jason D. Williams. 2014. The second dialog state tracking challenge. In Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL), pages 263–272, Philadelphia, PA, U.S.A. Association for Computational Linguistics.

Seokhwan Kim, Luis Fernando D'Haro, Rafael E Banchs, Jason D Williams, and Matthew Henderson. 2017. The fourth dialog state tracking challenge. Dialogues with Social Robots: Enablements, Analyses, and Evaluation, pages 435–449.

Yosuke Kishinami, Reina Akama, Shiki Sato, Ryoko Tokuhisa, Jun Suzuki, and Kentaro Inui. 2022. Target-guided open-domain conversation planning. In Proceedings of the 29th International Conference on Computational Linguistics, pages 660–668, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

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

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022a. Large language models are zero-shot reasoners. In Advances in Neural Information Processing Systems, volume 35, pages 22199–22213. Curran Associates, Inc.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022b. Large language models are zero-shot reasoners. Advances in neural information processing systems, 35:22199–22213.

Chuang Liu, Junzhuo Li, and Deyi Xiong. 2023. Tab-CQA: A tabular conversational question answering dataset on financial reports. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track), pages 196–207, Toronto, Canada. Association for Computational Linguistics.

Junhua Liu, Tan Yong Keat, Bin Fu, and Kwan Hui Lim. 2024. LARA: Linguistic-adaptive retrieval-augmentation for multi-turn intent classification. In Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track, pages 1096–1106, Miami, Florida, US. Association for Computational Linguistics.

Mehrad Moradshahi, Tianhao Shen, Kalika Bali, Monojit Choudhury, Gael de Chalendar, Anmol Goel, Sungkyun Kim, Prashant Kodali, Ponnurangam Kumaraguru, Nasredine Semmar, Sina Semnani, Jiwon Seo, Vivek Seshadri, Manish Shrivastava, Michael Sun, Aditya Yadavalli, Chaobin You, Deyi Xiong, and Monica Lam. 2023a. X-RiSAWOZ: High-quality end-to-end multilingual dialogue datasets and few-shot agents. In Findings of the Association for Computational Linguistics: ACL 2023, pages 2773–2794, Toronto, Canada. Association for Computational Linguistics.

Mehrad Moradshahi, Tianhao Shen, Kalika Bali, Monojit Choudhury, Gaël de Chalendar, Anmol Goel, Sungkyun Kim, Prashant Kodali, Ponnurangam Kumaraguru, Nasredine Semmar, et al. 2023b. X-risawoz: High-quality end-to-end multilingual dialogue datasets and few-shot agents. ArXiv preprint, abs/2306.17674.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. Advances in Neural Information Processing Systems, 35:27730–27744.

Jun Quan, Shian Zhang, Qian Cao, Zizhong Li, and Deyi Xiong. 2020. RiSAWOZ: A large-scale multi-domain Wizard-of-Oz dataset with rich semantic annotations for task-oriented dialogue modeling. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 930–940, Online. Association for Computational Linguistics.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, pages 8689–8696. AAAI Press.

Morgan Sandler, Hyesun Choung, Arun Ross, and Prabu David. 2024. A linguistic comparison between human and chatgpt-generated conversations. ArXiv preprint, abs/2401.16587.

Karin Sevegnani, David M. Howcroft, Ioannis Konstas, and Verena Rieser. 2021. OTTers: One-turn topic transitions for open-domain dialogue. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 2492–2504, Online. Association for Computational Linguistics.

Dan Shi, Chaobin You, Jiantao Huang, Taihao Li, and Deyi Xiong. 2024. Corecode: A common sense annotated dialogue dataset with benchmark tasks for chinese large language models. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, pages 18952–18960.

Noah Shinn, Beck Labash, and Ashwin Gopinath. 2023. Reflexion: an autonomous agent with dynamic memory and self-reflection. ArXiv preprint, abs/2303.11366.

Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Le Hou, Kevin Clark, Stephen Pfohl, Heather Cole-Lewis, Darlene Neal, et al. 2023. Towards expert-level medical question answering with large language models. ArXiv preprint, abs/2305.09617.

Jianheng Tang, Tiancheng Zhao, Chenyan Xiong, Xiaodan Liang, Eric Xing, and Zhiting Hu. 2019. Target-guided open-domain conversation. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 5624–5634, Florence, Italy. Association for Computational Linguistics.

Jian Wang, Yi Cheng, Dongding Lin, Chak Tou Leong, and Wenjie Li. 2023a. Target-oriented proactive dialogue systems with personalization: Problem formulation and dataset curation. ArXiv preprint, abs/2310.07397.

Jian Wang, Dongding Lin, and Wenjie Li. 2022a. Follow me: Conversation planning for target-driven recommendation dialogue systems. ArXiv preprint, abs/2208.03516.

Jian Wang, Dongding Lin, and Wenjie Li. 2023b. Dialogue planning via brownian bridge stochastic process for goal-directed proactive dialogue. ArXiv preprint, abs/2305.05290.

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

Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. 2022. Generating sequences by learning to self-correct. ArXiv preprint, abs/2211.00053.

Wenquan Wu, Zhen Guo, Xiangyang Zhou, Hua Wu, Xiyuan Zhang, Rongzhong Lian, and Haifeng Wang. 2019. Proactive human-machine conversation with explicit conversation goals. ArXiv preprint, abs/1906.05572.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models, may 2023. ArXiv preprint, abs/2305.10601.

Zihao Yi, Jiarui Ouyang, Yuwen Liu, Tianhao Liao, Zhe Xu, and Ying Shen. 2024. A survey on recent advances in llm-based multi-turn dialogue systems. arXiv preprint arXiv:2402.18013.

Chaobin You and Deyi Xiong. 2024. Tclnet: Turn-level contrastive learning network with reranking for dialogue state tracking. Knowledge-Based Systems, 302:112308.

Xiao Yu, Maximillian Chen, and Zhou Yu. 2023. Prompt-based monte-carlo tree search for goal-oriented dialogue policy planning. ArXiv preprint, abs/2305.13660.

Jun Zhang, Yan Yang, Chencai Chen, Liang He, and Zhou Yu. 2021. KERS: A knowledge-enhanced framework for recommendation dialog systems with multiple subgoals. In Findings of the Association for Computational Linguistics: EMNLP 2021, pages 1092–1101, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Shaowei Zhang and Deyi Xiong. 2025. BackMATH: Towards backward reasoning for solving math problems step by step. In Proceedings of the 31st International Conference on Computational Linguistics: Industry Track, pages 466–482, Abu Dhabi, UAE. Association for Computational Linguistics.

Tengxun Zhang, Hongfei Xu, Josef van Genabith, Deyi Xiong, and Hongying Zan. 2023. Napg: Non-autoregressive program generation for hybrid tabular-textual question answering. In CCF International Conference on Natural Language Processing and Chinese Computing, pages 591–603. Springer.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. 2022. Least-to-most prompting enables complex reasoning in large language models. ArXiv preprint, abs/2205.10625.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Long Li, Jialong Wu, Tiannan Wang, Shi Qiu, Jintian Zhang, Jing Chen, Ruipu Wu, Shuai Wang, et al. 2023. Agents: An open-source framework for autonomous language agents. ArXiv preprint, abs/2309.07870.

Qi Zhu, Kaili Huang, Zheng Zhang, Xiaoyan Zhu, and Minlie Huang. 2020. CrossWOZ: A large-scale Chinese cross-domain task-oriented dialogue dataset. Transactions of the Association for Computational Linguistics, 8:281–295.

Xinyu Zhu, Junjie Wang, Lin Zhang, Yuxiang Zhang, Ruyi Gan, Jiaxing Zhang, and Yujiu Yang. 2022. Solving math word problem via cooperative reasoning induced language models. ArXiv preprint, abs/2210.16257.

A Appendix

A.1 Task Definition

Task Definition for dialogue should include the definitions of fields $task\_profile$, $user\_profile$, $agent\_action$, and $user\_state$. The definitions of these fields are as follows:

- $task\_profile$: Task information for providing business content, accomplishment goals, and relevant background knowledge to Agents. The example is as follows:

```
{
    "task_goal": "Invite the user to a golf experience event",
    "success_mark": ["Agent.InformBookingSuccess"],
    "agent_identity": "Customer Service of ** Bank Wealth Center",
    "event_time": "1st to 31st of the current month",
    "event_location": "Shenzhen Golf Club",
    "...":"...",
    "other_knowledge": "Event Notification Document. Respectful
Bank Customers, We sincerely invite you to participate in the
upcoming golf experience event. Here are the detailed information
..."
}
```

Figure 7: Task profile prompt setting.

- $user\_profile$: The information about user business and personal information held by agents generally comes from the company's user management system. This information is used for identity verification or providing personalized services in conversations. The following is a example:

```
{
    "Name": "Li **",
    "customer_type": "Large Deposit",
    "Age": "30",
    "Occupation": "Executive of a Listed Company"
}
```

Figure 8: User profile prompt setting.

- $agent\_action$: Summary of key actions to be carried out during the process of agent dialogue, intended to guide and constrain the content of the agent's dialogue in accordance with business regulations. Typically corresponds to the node names in the SOP or the strategy names of proactive dialogue. Here are some examples:

```
{
    "agent_action": [
        "Start",
        "VerifyIdentity",
        "...",
        "AttemptPersuasion",
        "Chat",
        "OtherActions"
    ]
}
```

Figure 9: Agent actions setting.

- $user\_state$: The status of the task summary from the user's final response combined with the preceding dialogue, serves as a prompt for the agent to select the optimal next action. This typically corresponds to the user node in the SOP or the proactive dialogue state of the user. The following are examples:

```
{
    "user_state": [
        "NotThemselves",
        "IsThemselves",
        "ClearAgreement",
        "...",
        "DelayDecision",
        "Chat",
        "OtherIntentions"
    ]
}
```

Figure 10: User States setting.

A.2 SOP Definition

Standardized operating procedures (SOP) is a directed graph where vertexes are $agent\_action$ and $user\_state$, and the edges indicate the connections between these nodes. The SOP is established by business experts to standardize the essential business processing steps. Omitting or altering these steps can result in a violation or error. For instance, in the "Activate Credit Card Invitation" task, the agent must first "verify the user's identity" and confirm that the user is a "cardholder" before introducing the "activation activity." Conversely, steps that do not impact the business process, such as the user "asking questions" and the agent "resolving doubts," should not be included in the SOP. When actions are included in the SOP, they are referred to as controllable actions; otherwise, they are termed proactive actions. Figure 11 shows an example of SOP graph. The adjacency list representation of the SOP is shown in Figure 12.

A.3 Dataset Curation Details

A.3.1 Prompts of Role-Playing Framework

The prompts for step 2, 3, and 4 are shown in A.7.

A.3.2 Annotators and Annotations

I. Annotation Recruiter Selection Process. We recruited annotators from a Chinese university and

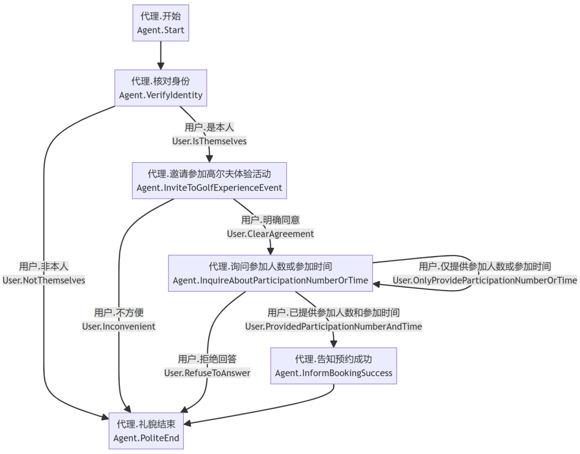


Figure 11: An example of SOP definition.

```
{
    "Agent.Start": [
        "Agent.VerifyIdentity"
    ],
    "Agent.VerifyIdentity": [
        "User.NotThemselves",
        "User.IsThemselves"
    ],
    "User.IsThemselves": [
        "Agent.InviteToGolf..."
    ],
    "...":"...",
    "User.ProvidedParticipation...": [
        "Agent.InformBookingSuccess"
    ],
    "Agent.InformBookingSuccess": [
        "Agent.PoliteEnd"
    ]
}
```

Figure 12: The adjacency list representation of the SOP.

renowned enterprises through a three-stage selection process, ultimately choosing seven qualified annotators. The specific recruitment process includes three steps:

- **Qualification Selection** Candidates must have a bachelor's degree or above, and background knowledge in NLP, and preference is given to those with annotation experience.
- **Trial Annotation** Candidates who pass the qualification selection are provided with uniform training. After the training, they perform trial annotations on a small-scale dataset, and the results are assessed for compliance with the annotation rules (for details, see A.3.3).
- **Diversity Selection** From those who pass the trial annotation, we select individuals from different university majors and professional departments to increase diversity.

II. Measures to Ensure Dialogue Quality and Consistency Implement Unified Training by integrating a Training Manual, Meeting Clarifications, and Examples. Conduct pilot annotations to identify and resolve issues during the trial annotation period. Timely communication, feedback, and discussion during the process, promptly supplement and update the training manual, and re-examine the annotated results. Additionally, every instance was assigned to three annotators for cross-annotation validation with an inter-annotator agreement (IAA).

III. Compensation. On average, the annotation time was 10.3 minutes per sample, and the compensation was $8 per hour.

A.3.3 Annotation Guidelines

All annotators used the annotation tool for marking, and a screenshot of the annotation tool is shown in Figure 13. The complete annotation guideline includes numerous definitions and examples, with detailed formatting as shown above A.1. Below is a brief explanation of the key annotation fields and important considerations.

Step 1 annotation guidelines. Task 1 requires annotating the task definition. The task should come from various domains in the real world, and the task process can be described with core steps using SOP. The task can be completed in the form of a dialogue between the agent and the user. Content to be annotated:

- **Task Profile:** Provide the agent with relevant business knowledge about this dialogue task, which must include "agent_identity", "agent_goal" and "success_mark". Additional necessary business knowledge can be supplemented.
- **User Profile:** Provide the agent with information about the user, who is the subject of this task dialogue, such as their name, etc.
- **Agent Action:** The dialogue actions that the agent can choose to facilitate the completion of the task.
- **User State:** The task status is achieved based on the information provided by the user during the dialogue process.

Step 2 annotation guidelines. Task 2 requires the creation of SOP for the tasks defined in Task 1. The following should be annotated:

- **SOP Vertexes:** Selected from agent actions and user states, used to define the core SOP diagram that needs to be followed.

Figure 13: Screenshot of the annotation tool.

- **SOP:** The interconnections between the nodes in the SOP diagram. It starts with "Agent.Start" and ends with "Agent.PoliteEnd", meaning that the leaf nodes have only one.

Step 3 annotation guidelines. The dialogue path is a sequence of interactive actions generated by the LLM based on the SOP, representing a specific user completing a conversation in a particular setting. It is necessary to check whether the dialogue actions in the path follow the order in the core SOP. At the same time, for agent actions that are not part of the SOP, it is necessary to check whether they reflect the agent's proactivity, that is, whether the agent is striving to guide the conversation toward the task goal.

Step 4 annotation guidelines. Annotators need to correct the dialogue content and dialogue actions (task status) to ensure they are consistent with the background knowledge provided for the task, comply with the constraints of the SOP, and adhere to common sense. Additionally, they should correct the grammatical correctness and fluency of the dialogue.

A.3.4 List of Tasks in 32 Domains

All domains and tasks are shown in Table 7.

A.4 Experiment Details

A.4.1 Offline Planner: Task 1

The prompt for Direct Adjacency List (DAL) is shown in A.7. TCoT first-step: the prompt for LLM to describe each vertex and its child vertices in natural language is shown in A.7. TCoT second-step: the prompt for LLM to translate the description into an adjacency list in JSON format is shown in A.7

A.4.2 Online Planner: Task 2

In MCTS, the number of generated actions M is set to 3, depth limit L is set to 8, the number of roll-outs N is set to 3, and exploration weight w is set to 1. To maintain a search scale similar to MCTS, in ToT, the number of generated actions M is set to 3, depth limit L is set to 8. The prompt for ToT sampling actions, generating responses, and predicting the user state is exactly the same as that for MCTS. The prompts for CoT, CoT+SOP, MCTS, and ToT are shown in A.7.

A.4.3 Supervised Fine-Tuning

All open-source models' experiments were completed on 4 Nvidia A800 GPUs. For SFT, we fine-tuned all parameters using 5 epochs, 50 warm steps, 128 batch size, and 1e-6 learning rate.

A.4.4 Generation Parameters

In the experiment of task 1, the temperature was uniformly set to 0.1 and the top-p was set to 0.1.

Domains	Tasks
scholarism	conference invitation
courier	delayed package handling
bank	activate bank card, agent large transaction inquiry, financial product sales, golf invitation, loan followup
white goods	repair appointment, installation appointment
shopping	sams club member day invitation, redeem promotion, take out order
education	online market
workplace	apply for work card
photo studio	photo appointment
hospital	vaccine inform, appointment
airport	check in
cosmetology	product follow up, member day
household	unblocking pipeline, moving appointment, property fee deposit, recycling appointment
restaurant	private room booking, place a food order
cinema	movie ticket purchase
pet	complain consult, adoption facilitation
hotel	check in
entertainment	ktv complain consult
gym	private tutoring, swimming pass promotion
car insurance	sales promotion
community	competition, lost and found
library	borrow book
health	blood pressure monitoring
telecom	activate package, sim card upgrade promotion, broadband upgrade
domestic service	complain
school	home visit appointment, reissue student card, archive uery
tourism	booking, hot spring promotion
real estate	event invitation
internet	broadband repair phone support
glasses	fitting
computer	repair appointment
account	password recovery
survey	membership reward

Table 7: Domains and tasks Details.

For task 2, the temperature was uniformly set to 1 and the top-p was set to 0.95.

A.5 Human Evaluation Details

We continue to employ the annotators recruited for dataset construction to complete the evaluation of different models in dialogue tests. Let H denote the annotated preceding dialogue context, and let r represent the model's agent response to H. The specific evaluations include:

Controllability Accuracy If the content of r originates from the SOP, then r is added to a controllable response set R_{ctrl}. Furthermore, if r correctly adheres to the SOP procedural constraints, then r is added to a correctly controllable response set $R_{\text{ctrl\_right}}$. Finally, the controllability accuracy acc_{ctrl} is defined as:

$$acc_{\text{ctrl}} = \frac{\text{count}(R_{\text{ctrl\_right}})}{\text{count}(R_{\text{ctrl}})} \tag{7}$$

Proactivity Accuracy If the content of r originates not from the SOP, then r is added to a proactive response set R_{pact}. Furthermore, if r responds correctly to the user and effectively guides the conversation towards achieving the goal, then r is added to a correctly proactive response set $R_{\text{pact\_right}}$. Finally, the proactivity accuracy acc_{pact} is defined as:

$$acc_{\text{pact}} = \frac{\text{count}(R_{\text{pact\_right}})}{\text{count}(R_{\text{pact}})} \tag{8}$$

Knowledge Accuracy If r contains business or user information from the task, it will be added to the knowledge response set R_{know}. In addition, if r contains accurate information, free of errors or hallucinations, then r will be added to the correct knowledge response set $(r_{\text{know\_right}})$.Finally, the knowledge accuracy acc_{know} is defined as:

$$acc_{\text{know}} = \frac{\text{count}(R_{\text{know\_right}})}{\text{count}(R_{\text{know}})} \tag{9}$$

Goal Success Rate If the model's responses in each turn contain no controllability errors and knowledge errors, and the dialogue ultimately reaches the goal position, the dialogue is considered goal successful. The goal success rate is the proportion of successful dialogues to the total number of dialogues.

Logical Coherence Score We evaluate the logical coherence of the model by breaking down the scoring points across multiple dimensions.

- Response Logic: Whether the model misunderstands the user's intent and responds with content that is completely unrelated or incorrect to the intent. If there is an error, score 0; otherwise, score 0.4.

- Context Consistency: Whether the model's response content contradicts the dialogue history. If there is a contradiction, score 0; otherwise, score 0.3.

- Common Sense: If the model's response contradicts common sense, score 0 else 0.3.

A.6 Example of Dialogue

Listing 1: An Example of Dialogue

```
Agent Action: Greet
Agent Response: Hi, this is the customer service center of the
  ↪ ** Bank Credit Card Department.

User Response: Hi!
User State: Greet

Agent Action: VerifyIdentity
Agent Response: Hi, am I speaking with Ms. Hao **?

User Response: Yes, that's right.
User State: Cardholder

Agent Action: IntroduceActivationActivities.
Agent Response: Ms. Hao, thank you for applying for our
  ↪ platinum credit card. To reward our new customers, we
  ↪ currently have an activation promotion that offers a
  ↪ 12\% discount on fuel for one year. Once you activate
  ↪ your credit card, you can enjoy this benefit.

User Response: That sounds great, but I've never used a credit
  ↪ card before and I'm a bit worried it might be
  ↪ troublesome.
User State: WorryAndDoubt

Agent Action: RelieveDoubts
Agent Response: Ms. Hao, you don't have to worry at all.
  ↪ Activating a credit card is very simple. You can choose
  ↪ to handle it at the bank counter, or you can activate it
  ↪ through the mobile banking APP or by calling our
  ↪ credit card customer service hotline for self-activation.
  ↪ The whole process is very convenient and fast.

User Response: Oh, I see. That's good to know.
User State: HabitualResponseAndContinue

Agent Action: AttemptPersuasion
Agent Response: Yes, and after activation, you can also set a
  ↪ transaction password to ensure the security of your card
  ↪  usage. Our platinum credit card is free of annual fees
  ↪ for the first year, and you can waive the annual fee by
  ↪ swiping the card three times a year. This is not only safe
  ↪  and convenient but also comes with many promotional
  ↪ activities.

User Response: That sounds like a great deal, please activate it
  ↪  for me.
User State: Activate

Agent Action: SetPassword
Agent Response: Thank you for your trust. To ensure the
  ↪ security of your card transactions, please set a 6-digit
  ↪ transaction password.

User Response: Sure, my transaction password is ******.
User State: Success

Agent Action: Thank
Agent Response: Ms. Hao, your credit card has been
  ↪ successfully activated, and the transaction password has
  ↪  been set. You can now use your credit card normally.

User Response: Thank you, it's really convenient.
User State: Thank

Agent Action: Goodbye
Agent Response: You're welcome, Ms. Hao. If you have any
  ↪ questions, you can call our customer service hotline at
  ↪ any time. I wish you a happy life, goodbye!

User Response: Okay, goodbye!
User State: Goodbye
```

A.7 Prompt Details

Prompting LLM to predict SOP's Direct Adjacency List (DAL).

You are a professional business manager, having accumulated extensive knowledge of business processes and being adept at conversing with clients to complete business tasks.

Now, please create the standard operating procedure (SOP) diagram for this task based on the following "Task Profile". All vertexes of this SOP diagram are as follows "SOP Vertexes", and you need to output the list of adjacent for each vertex.

Please strictly follow the requirements below:
1. The output format should be in the form of a JSON string array.
2. The JSON fields must include all nodes. If a vertex has no adjacent vertexes, represent it with [].

Task Profile

```
{
   "agent_identity": "** Bank Credit Card Center Customer
 ↪  Service",
   "task_goal": "To persuade the user to explicitly agree
 ↪  to the credit card",
   "success\_mark": [
       "User.ClearAgreement"
       ],
   "...":"...",
   "other_knowledge": "Credit Card Opening Guide I. Card
 ↪  Activation ..."
}
```

SOP Vertexes

```
[
    "Agent.Start",
    "Agent.VerifyIdentity",
    "...",
    "Agent.PoliteEnd",
    "Agent.NotifyActivationSuccess"
]
```

The adjacency list of all vertexes:

Generation Output:

```
{
 "Agent.Start":[
    "Agent.VerifyIdentity"
    ],
 "Agent.VerifyIdentity":[
     "User.Cardholder",
     "User.NonCardholder"
     ],
 "...":[
    "..."
    ],
 "User.ClearRejection":[
    "Agent.PoliteEnd"
    ],
 "Agent.PoliteEnd":[],
 "Agent.NotifyActivationSuccess":[
    "Agent.PoliteEnd"
    ]
}
```

The prompt of TCoT to translate the description into an adjacency list.

You are a seasoned business representative, having accumulated a wealth of business process knowledge, and are adept at conversing with customers to complete transactions. Now please formulate the Standard Operating Procedure (SOP) diagram for this task.

You can refer to the following "Task Knowledge", but it may not be comprehensive or accurate, and you do not need to strictly follow it. All nodes of the SOP diagram are as follows "SOP Diagram Nodes", and you need to output the list of adjacent nodes for each node, that is, the adjacency list of the entire SOP diagram.

Hint

1. The SOP diagram starts with 'Agent.Start' and ends with 'Agent.PoliteEnd'.
2. The subsequent nodes of the "User" node are usually "Agent" nodes, and the subsequent nodes of the "Agent" node mostly require a response from the "User" node. For some actions that do not require a response, the subsequent node of "Agent" can also be "Agent".
3. If there is a 'VerifyIdentity' node in the task, it indicates that only the person themselves can continue, and the conversation ends if it is not the person themselves.
4. The JSON fields must include all nodes.
5. Please strictly follow output format, for example:

```
{
    "Agent.Start": [
        "Agent.Node1"
        ],
    "Agent.Node1": [
        "User.Node1",
        "User.Node2"
        ],
    "...": ["..."]
}
```

Task Knowledge
After 'Agent.Start', the first step should be 'Agent.VerifyIdentity'. After confirming the identity, it could be 'User.IsThemselves' or 'User.NotThemselves'. If not the person, the agent will 'Agent.PoliteEnd'. If the user is the person, then the agent will introduce the event, that is 'Agent.InviteToGolfExperienceEvent'...

SOP Diagram Nodes ["Agent.Start", "Agent.VerifyIdentity", ...]

Adjacency list for all nodes:

Generation Output:

```
{
    "Agent.Start":[
        "Agent.VerifyIdentity"
        ],
    "...":["..."]
}
```

The prompt of TCoT to describe the adjacency relationship between vertexes in natural language.

You are a professional business manager, having accumulated a wealth of business process knowledge and being adept at conversing with customers to complete transactions.

Now, based on the vertexes of the Standard Operating Procedure (SOP) diagram, please analyze all the vertexes. Write a passage analyzing the entire process of interaction and communication between the 'User' and 'Agent', covering all vertexes and all possible scenarios, such as what happens if the user 'Agree' or 'Disagree'.

Here is an example:
SOP vertexes:

['Agent.Start', 'Agent.VerifyIdentity', 'User.IsThemselves', 'Agent.IntroduceGiftDelivery', 'User.NotThemselves', 'User.ClearAgreement', 'Agent.AskForDeliveryAddress', 'User.ProvideDeliveryAddress', 'User.DoNotProvideDeliveryAddress', 'User.ClearRejection', 'Agent.PoliteEnd', 'Agent.NotifyRegistrationSuccess']

Start analysis:
After 'Agent.Start', the first step should be 'Agent.VerifyIdentity'. After confirming the identity, it could be 'User.IsThemselves' or 'User.NotThemselves'. If not the person, the agent will 'Agent.PoliteEnd'. If the user is the person, the agent will introduce, that is 'Agent.IntroduceGiftDelivery'. If 'User.ClearAgreement', the process can continue. Since it is a gift delivery, after the user agrees, 'Agent.AskForDeliveryAddress', if 'User.ProvideDeliveryAddress' then the agent 'Agent.NotifyRegistrationSuccess', if 'User.DoNotProvideDeliveryAddress', the agent may ask again 'Agent.AskForDeliveryAddress', or directly 'Agent.PoliteEnd'. After 'Agent.NotifyRegistrationSuccess', it can 'Agent.PoliteEnd'.

SOP vertexes:
["Agent.Start", "Agent.VerifyIdentity", ...,"Agent.PoliteEnd"]

Start analysis:

Generation Output:
After 'Agent.Start', the first step should be 'Agent.VerifyIdentity'. After confirming the identity, it could be 'User.IsThemselves' or 'User.NotThemselves'. If not the person, the agent will 'Agent.PoliteEnd'. If the user is the person, then the agent will introduce the event, that is 'Agent.InviteToGolfExperienceEvent'. If ...

Prompting LLM to generate full dialogue paths.

You are a professional business manager, assisting customers in handling their business through dialogue.
Based on the provided “task_profile”, “user_profile” and “sop_adjacency_list”, Please insert more “user_state” and “agent_actions” into the above “Main Dialogue Path” to enrich the possible reactions of the user and the corresponding decision-making actions that the agent takes in response to the user’s state, forming a full dialogue path.
Below is the related information:

```
{
    "task_profile": {"...":"..."},
    "user_profile": {"...":"..."},
    "sop_adjacency_list": {"...":["..."]},
    "agent_action": [
        "Agent.VerifyIdentity",
        "..."
    ],
    "user_state": [
        "User.Cardholder",
        "..."
    ]
}
```

Main Dialogue Path:
[“Agent.VerifyIdentity”, “User.Cardholder”...]

Please follow the requirements below strictly:

1. Full dialogue paths should start with “Agent.Greeting” and end with “User.Ending”.
2. Require “Agent Action” and “User State” to be spoken alternately, with each occurrence representing a round of dialogue. Please add a “–” after each round of dialogue to separate the previous and next rounds. For example, [“Agent.Greeting”, “User.Greeting”, “–”, “Agent.IntroduceActivity”,...].
3. Don’t delete nodes or adjust the nodes’ order in the “Main Dialogue Path”.
4. Please only insert 2 to 5 reasonable rounds of dialogue.
5. The inserted nodes can only be selected from the “user_state” and “agent_action” defined in the above text, and do not create new names.
6. When the preceding node comes from “sop”, please try to choose one from the subsequent nodes of the current node in the adjacency list.
7. The output format should be a JSON string array.

Supplemented full dialogue paths in JSON format:

Generation Output:

```
[
    "Agent.Greeting",
    "User.Greeting",
    "--",
    "Agent.VerifyIdentity",
    "User.Cardholder",
    "--",
    "Agent.IntroduceActivity"
    "..."
]
```

Prompting LLM to generate dialogue according to full dialogue paths.

You are a professional business manager, assisting customers with their business through dialogues.

Now, due to the need for training new business representatives, you are asked to act as a scriptwriter. Based on the “task_profile” and “user_profile” provided below, and the “Full Dialogue Path” between the agent and the user for this task, write dialogue scripts that fit the roles of both parties.
Here are the task-related information:

```
{
    "task_profile":{
        "agent_identity": "** Bank Credit Card Center
        ↪ Customer Service",
        "...",
        "other_knowledge": "Credit Card Activation Guide I.
        ↪ Card Activation ..."
    },
    "user_profile": {
        "Name": "**",
        "Gender": "Male",
        "...",
        "CardID":"****"
    }
}
```

Full Dialogue Paths:
[“Agent.Greeting”, “User.Greeting”, “Agent.VerifyIdentity”, “User.Cardholder”, ...]

Please write dialogue scripts for the “Full Dialogue Paths” based on the task-related information, forming a complete conversation.

Please strictly follow the requirements below:

1.Please directly append the “script” you have written after the name of each node in the “Complete Dialogue Path”, separated by the “|” symbol. For example: “User.Consider|I understand, I will think about it.”

2.Do not add, delete, or change any original node.

3.The complete output format should be consistent with the “Complete Dialogue Path”, that is, output as a JSON array of strings, such as [“Agent.PoliteEnd|Thank you for your cooperation, I wish you a happy life!”, “User.Ending|Okay, goodbye!”]

Please output the “Complete Dialogue Path” after adding the dialogue scripts:

Generation Output:

```
[
    "Agent.Greeting|Hello, this is the credit card center of
    ↪ ** Bank.",
    "User.Greeting|Hello",
    "Agent.VerifyIdentity|Hi, may I ask if you are Ms. **?",
    "User.Cardholder| Yes, this is ** speaking. ",
    "Agent.IntroduceActivationActivity|That's correct, Mr.*
    ↪ we have noticed that your platinum credit card has
    ↪ not been activated yet. To express our gratitude for
    ↪ your support, we currently have a promotion where
    ↪ you can enjoy a 12\% discount on fuel for one year
    ↪ just by activating your card.",
    "..."
]
```

The prompt for sampling candidate agent action to expand in SGM.

Given the following "user_profile", "task_profile" and "Dialogue Context". Please select the next round of agent action from the "Optional Agent Actions" for the agent that can best encourage the user to complete this dialogue task. When the task is completed, please actively choose "Agent.PoliteEnd" to end the conversation. The name of the agent action must be exactly the same as those in the candidate list.

```
{
    "task_profile":{
        "agent_identity": "** Bank Credit Card Center
        ↪  Customer Service",
        "...",
        "other_knowledge": "Credit Card Activation Guide I.
        ↪  Card Activation ..."
    },
    "user_profile": {
        "Name": "**",
        "Gender": "Male",
        "...",
        "CardID":"****"
    }
}
```

Optional Agent Actions:
["Start", "VerifyIdentity", "IntroduceActivationActivity", ...,"RelieveDoubts", "AttemptPersuasion", "Chat", "OtherActions"]

Dialogue Context:
Agent Action: Greeting
Agent Response: Hello, this is the credit card center of ** Bank.
User Response: Hello!
User State: Greeting
Agent Action: VerifyIdentity
Agent Response: Hi, may I ask if you are Ms. **?
User Response: Yes, this is ** speaking.

Dialogue SOP:
Based on the standard procedure for this task, please prioritize the following agent actions:
["IntroduceActivationActivity"]

Start with "Analysis:" and deduct step by step the best agent actions that should be used by the subsequent dialogue agent, concluding with an output similar to "Therefore, the best agent action is: Greeting", with a colon required before the name of the agent action.

Generation Output:

Analysis: According to the previous conversation, the user confirms their identity, indicating that the user is the person the agent needs to verify. Following the task process prompts, the next step is to invite them to participate in the activation activity, which aligns with the conversation task logic.

Therefore, the best agent action is: IntroduceActivationActivity

The prompt generates an agent response for MCTS's simulation.

Given the "user_profile", "task_profile", and "Dialogue Context" for the dialogue task below, please generate an appropriate agent response by following the chosen agent action. Start your response with a new line and "Agent Response:".

```
{
    "task_profile":{
        "agent_identity": "** Bank Credit Card Center
        ↪  Customer Service",
        "...",
        "other_knowledge": "Credit Card Activation Guide I.
        ↪  Card Activation ..."
    },
    "user_profile": {
        "Name": "**",
        "Gender": "Male",
        "...",
        "CardID":"****"
    }
}
```

Dialogue Context:
Agent Action: Greeting
Agent Response: Hello, this is the credit card center of ** Bank.
User Response: Hello!
User State: Greeting
Agent Action: VerifyIdentity
Agent Response: Hi, may I ask if you are Ms. **?
User Response: Yes, this is ** speaking.
User State: Cardholder
Agent Action: IntroduceActivationActivity

Generation Output:
Agent Response: Hi Mr. **, this is the credit card center of ** Bank. I noticed you haven't activated your Platinum credit card yet, and I wanted to thank you for being a valued customer. Right now, we're running a special promotion where you can get a 12% discount on fuel for an entire year just by activating your card. It's a great way to save on your everyday expenses. Would you like me to help you get started with the activation today?

The prompt for MCTS to get a reward score.

You are a business expert manager at a company in the financial sector. The company's business agent is contacting the customer below to handle affairs, see "Dialogue Context" for the content of the conversation. Now, please carefully read follow information and judge whether the agent's chosen "Agent action" conforms to the task process (sop) or is conducive to guiding the user toward the direction of task success. If it meets the requirements, it is 1; otherwise, it is 0.

```
{
    "task_profile":{
        "agent_identity": "** Bank Credit Card Center
        ↪ Customer Service",
        "...",
        "other_knowledge": "Credit Card Activation Guide I.
        ↪ Card Activation ..."
    },
    "user_profile": {
        "Name": "**",
        "Gender": "Male",
        "...",
        "CardID":"****"
    },
    "sop_adjacency_list": {"...":["..."]},
    "agent_action": [
        "Agent.VerifyIdentity",
        "..."
    ]
}
```

Dialogue Context:
Agent Action: Greeting
Agent Response: Hello, this is the credit card center of ** Bank.
User Response: Hello!
User State: Greeting
Agent Action: VerifyIdentity
Agent Response: Hi, may I ask if you are Ms. **?
User Response: Yes, this is ** speaking.
User State: Cardholder
Agent Action: IntroduceActivationActivity

Start your analysis with "Analysis:" and after your conclusion, summarize with "Therefore, the answer is: 1" or "Therefore, the answer is: 0".

Generation Output:
Analysis: According to the task knowledge, the objective of the agent's task is to invite users to participate in credit card activation event. The agent action chosen is IntroduceActivationActivity, which aligns with the task objective. Therefore, it is consistent with the above handling process and the best action within the optional agent actions.

Therefore, the answer is: 1

Prompting LLM to generate user state.

You are an agent specializing in lifestyle services. Given the following "user_profile", "task_profile', and "Dialogue Context", please select an option from the "Optional User State" that best reflects the user's current task status in the dialogue context.

Start with "User State:" and the state name must be exactly the same as one in the list.

```
{
    "task_profile":{
        "agent_identity": "** Bank Credit Card Center
        ↪ Customer Service",
        "...",
        "other_knowledge": "Credit Card Activation Guide I.
        ↪ Card Activation ..."
    },
    "user_profile": {
        "Name": "**",
        "Gender": "Male",
        "...",
        "CardID":"****"
    }
}
```

Dialogue Context: Agent Action: Greeting
Agent Response: Hello, this is the credit card center of ** Bank.
User Response: Hello!
User State: Greeting
Agent Action: VerifyIdentity
Agent Response: Hi, may I ask if you are Ms. **?
User Response: Yes, this is ** speaking.

Optional User State:
["Greet", "Cardholder","..."]

Generation Output:
User State: Cardholder

The prompt for CoT.

You are a business expert agent in the financial field. Given the following "user_profile", "task_profile', and "Dialogue Context", please output the following three items **step by step**:
1. Please select an option from "Optional User State" that best reflects the user's current task status in the dialogue above, and start with "User State:". Its name must match exactly with one from the optional list.
2. Please select an option from "Optional Agent Action" that best reflects the agent's next round of action that can prompt the user to complete the dialogue task. Start with a new line and "Agent Action:". Its name must match exactly with the one on the candidate list.
3. Please generate an appropriate Agent Response combined with the agent's chosen dialogue action. Start with a new line and "Agent Response:"

```
{
    "task_profile": {"...":"..."},
    "user_profile": {"...":"..."},
    "agent_action": [
        "Agent.VerifyIdentity",
        "..."
    ],
    "user_state": [
        "User.Cardholder",
        "..."
    ]
}
```

Dialogue Context:
Agent Action: Greeting
Agent Response: Hello, this is the credit card center of ** Bank.
User Response: Hello!

Generation Output:
User State: Greeting
Agent Action: VerifyIdentity
Agent Response: Hi, may I ask if you are Ms. **?

The prompt for CoT+SOP.

You are a business expert agent in the financial field. Given the following "user_profile", "task_profile', and "Dialogue Context", please output the following three items **step by step**:
1. Please select an option from "Optional User State" that best reflects the user's current task status in the dialogue above, and start with "User State:". Its name must match exactly with one from the optional list.
2. Please select an option from "Optional Agent Action" that best reflects the agent's next round of action that can prompt the user to complete the dialogue task. Start with a new line and "Agent Action:". Its name must match exactly with the one on the candidate list.
3. Please generate an appropriate Agent Response combined with the agent's chosen dialogue action. Start with a new line and "Agent Response:"

```
{
    "task_profile": {"...":"..."},
    "user_profile": {"...":"..."},
    "agent_action": [
        "Agent.VerifyIdentity",
        "..."
    ],
    "user_state": [
        "User.Cardholder",
        "..."
    ]
}
```

Dialogue Context:
Agent Action: Greeting
Agent Response: Hello, this is the credit card center of ** Bank.
User Response: Hello!

Dialogue SOP:
Based on the standard process of handling this task and the dialogue context, please prioritize the following agent action that meets the Standard Operating Procedures (SOP):
["VerifyIdentity"]

Generation Output:
User State: Greeting
Agent Action: VerifyIdentity
Agent Response: Hi, may I ask if you are Ms. **?

Algorithm 1 Online Planning for LLM Dialogue Agents with MCTS

Require: Initial dialogue state d_0, state transition probability function p_θ, reward function r_θ, action generator a_ϕ
Require: Number of expand actions M, depth limit L, number of roll-outs N, and exploration weight w
Initialize memory of actions $A : D \to A$, children $c : D \times A \to D$ and rewards $r : D \times A \to \mathbb{R}$
Initialize the state-action value function $Q : D \times A \to \mathbb{R}$ and visit counter $N : D \to \mathbb{N}$
for $k \leftarrow 0, \ldots, N-1$ **do**
    $t \leftarrow 0$
    while $N(d_t) > 0$ **do** ▷ {Selection}
        $N(d_t) \leftarrow N(d_t) + 1$
        $a_t \leftarrow \arg\max_{p \in A(d_t)} \left[Q(d_t, p) + w\sqrt{\frac{\ln N(d_t)}{N(c(d_t,p))}} \right]$
        $r_t = r(d_t, a_t), d_{t+1} \leftarrow c(d_t, a_t)$
        $t \leftarrow t + 1$
    end while
    while d_t is not a terminal state $\wedge t \leq L$ **do** ▷ Expansion
        for $i \leftarrow 1, \ldots, M$ **do**
            Sample $a_t^{(i)} \sim a_\phi(p|d_t), d_{t+1}^{(i)} \sim p_\theta(d_t, a_t^{(i)}), r_t^{(i)} \sim r_\theta(d_t, a_t^{(i)})$
            Update $A(d_t) \leftarrow \left\{ a_t^{(i)} \right\}_{i=1}^{d}, c(d_t, a_t^{(i)}) \leftarrow d_{t+1}^{(i)}, r(d_t, a_t) \leftarrow r_t^{(i)}$
        end for
        $a_{t+1} \leftarrow \arg\max_{a \in A(d_t)} r(d_t, a_t)$ ▷ Simulation
        $r_t \leftarrow r(d_t, a_t), d_{t+1} \leftarrow c(d_t, a_t)$
        $t \leftarrow t + 1$
    end while
    for $t' \leftarrow t, \ldots, 0$ **do** ▷ Back propagation
        Update $Q(d_{t'}, a_{t'})$ with $\{r_{t'}, r_{t'+1}, \ldots, r_t\}$
    end for
end for

A.8 Algorithm Details

Algorithm 1 presents the pseudocode for a Monte Carlo Tree Search (MCTS) planning algorithm designed for large language model (LLM) dialogue agents. The algorithm aims to improve dialogue policy quality by effectively exploring and selecting dialogue actions through simulation and backpropagation mechanisms.