Asking to Clarify: Bridging Ambiguity in Tool-Agent-User Interactions

Anonymous ACL submission

Abstract

Tool-using language agents often struggle under underspecified user instructions due to uncertainty about the user's true goal. We propose a plug-and-play Clarifier-augmented training and evaluation framework built on τ-Bench, in which a dedicated Clarifier is triggered after tool calls to decide whether and how to ask follow-up questions, without modifying the agent's internal policy. We compare supervised fine-tuning (SFT), direct preference optimization (DPO), and a user-goal-driven groupwise preference optimization (GRPO) method that trains the Clarifier using goal-aware preference signals conditioned on the ground-truth user requirement. Experiments across domains and agent backbones show that learned clarification consistently improves task success, with user-goal driven GRPO achieving among the strongest cross-domain generalization results, including robust gains on the out-of-distribution airline domain. In cross-agent evaluations over six agent backbones, the learned Clarifier improves success rate by 5.2% on average while increasing average interaction steps by only 0.1, demonstrating the effectiveness of a learned, plug-and-play clarification policy with minimal interaction overhead. Our code and data are available at the anonymous repository: `https://anonymous.4open.science/r/submission-E65F`.

1 Introduction

Large language model (LLM) agents have recently demonstrated strong capabilities in tool use and multi-step decision making across a wide range of real-world applications (Qu et al., 2025; Yao et al., 2022; Wölflein et al., 2025). By interacting with external tools, such agents are able to execute complex tasks that require planning, reasoning, and iterative feedback (Wölflein et al., 2025). However, in practical interactive settings, user instructions are often *underspecified* or ambiguous, leaving critical details of the user's true goal implicit (Yehudai et al., 2025; Qi et al., 2025). Such fuzzy user requirements pose a fundamental challenge for tool-using agents: premature or incorrect tool calls can lead to irreversible errors, brittle execution trajectories, and task failure (Wang et al., 2024a).

Clarifier Model	Retail (%)	Airline (%)	Average (%)
No Clarifier	16.5	14.3	15.4
Qwen3-8B	15.7 (-0.8)	8.0 (-6.3)	11.9 (-3.5)
Qwen3-14B	16.5 (0.0)	16.0 (+1.7)	16.3 (+0.9)
Qwen3-30B	15.7 (-0.8)	**19.0** (+4.7)	17.4 (+2.0)
DeepSeek-R1	14.8 (-1.7)	**19.0** (+4.7)	16.9 (+1.5)
DeepSeek-V3.1	**19.1** (+2.6)	17.0 (+2.7)	**18.1** (+2.7)

Table 1: **Impact of the Clarifier module on τ-Bench `retail` and `airline` performance.** Evaluations utilize Qwen3-8B as both the backbone agent and the simulated user. The row "No Clarifier" denotes the baseline. The reported values are Pass@1 (%), with absolute changes relative to the baseline shown in parentheses. The **Bold** indicates the best performance in each category.

A natural response to underspecified instructions is to ask clarification questions (Clark, 1996; Traum and Larsson, 2003; Zou et al., 2023). Human assistants routinely solicit missing information before acting, yet this capability remains poorly developed in modern LLM-based agents (Rao and Daumé III, 2018). Most LLMs are trained primarily for answering questions rather than asking them, and *consequently*, existing standard post-training pipelines offer limited supervision on when and how an agent should request additional information (Fu et al., 2020; Scialom et al., 2019; Ouyang et al., 2022; Ziegler et al., 2019; Zhang et al., 2024). As a result, clarification behavior in current agent systems is often heuristic, implicit, or entangled with the agent's internal reasoning process, making it difficult to control, optimize, or generalize across domains and agent backbones (Suri et al., 2025).

Table 1 evaluates the impact of different clarifier models on an extended version of τ-Bench (Yao et al., 2024), augmented with an explicit Clarifier

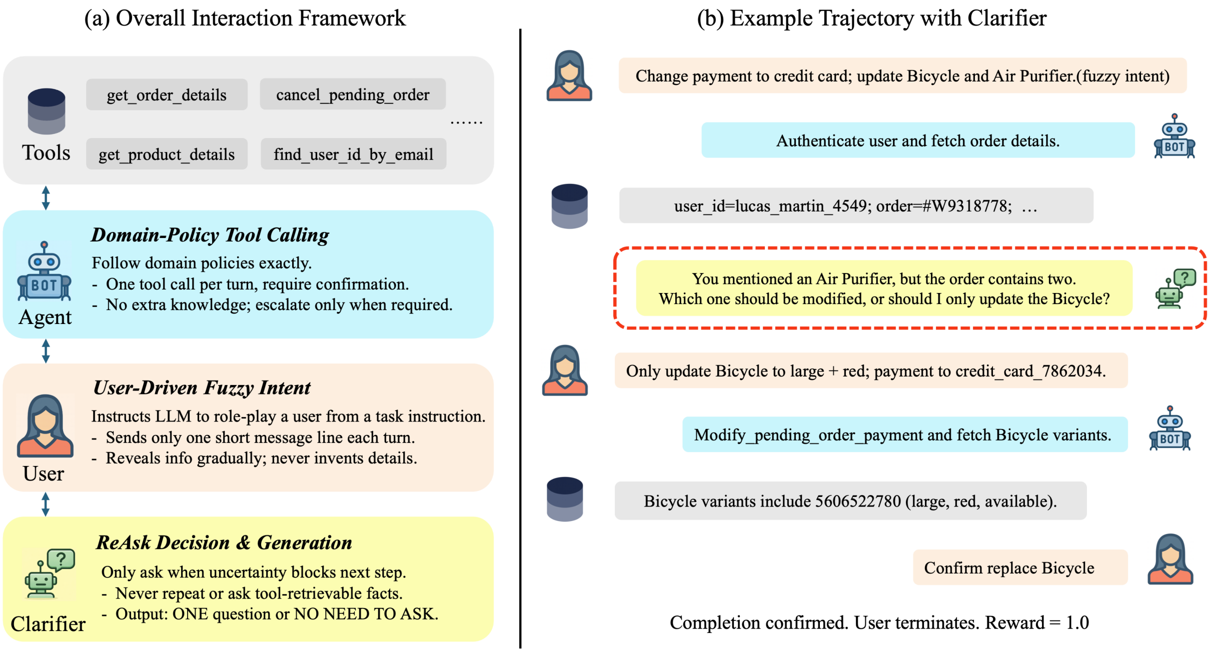


Figure 1: **Interaction framework of τ-bench augmented with a Clarifier model. (a)** We extend the standard τ-bench setup by introducing a *Clarifier* that triggers after tool calls when the agent has not clarified yet may still lack information for a valid next action. This yields a three-part prompting architecture: (1) *Domain-Policy Tool Calling* governing policy-compliant tool use; (2) *User-Driven Fuzzy Intent* modeling partially specified user goals; and (3) *ReAsk Decision & Generation* implementing uncertainty detection and targeted question generation. **(b)** Example τ-airline trajectory: when key parameters are missing from the user's fuzzy request, the Clarifier issues a concise follow-up question (highlighted), preventing policy violations and enabling the agent to proceed correctly.

module (Figure 1). This benchmark requires agents to infer latent user goals through multi-turn interactions under underspecified instructions. While introducing clarification generally unlocks higher success rates in ambiguous scenarios, effective questioning currently comes at a high cost. Significant gains are primarily observed in massive proprietary models (e.g., Deepseek V3.1), whereas clarification attempts by smaller off-the-shelf models often yield negligible gains or even performance drops. This disparity highlights the value of learning to ask: through explicit optimization, efficient models can rival the clarification capabilities of massive systems, thereby making robust questioning achievable at a fraction of the cost.

Motivated by this observation, we focus on learning clarification behavior specifically for tool-using agents. From a Bayesian perspective, the user's true goal is a latent variable partially observed through dialogue and tool feedback, and clarification questions serve as information-gathering actions that reduce this uncertainty before irreversible tool calls. Building on this perspective, we present the first approach to explicitly train a dedicated clarification model for tool-using agents, focusing on asking follow-up questions to resolve fuzzy user requirements during task execution. We introduce a Clarifier-augmented training and evaluation framework built on τ-Bench, in which a dedicated Clarifier module is triggered after tool calls to decide whether and how to ask a follow-up question. Crucially, the Clarifier is designed to be plug-and-play: it operates independently of the agent's internal policy and can be paired with different agent backbones without retraining the agent itself, enabling systematic study of clarification strategies across agents and domains. To robustly equip this module with strategic questioning capabilities, we investigate supervised fine-tuning (SFT) (Ouyang et al., 2022), direct preference optimization (DPO) (Rafailov et al., 2023), and a user-goal-driven group-wise preference optimization (GRPO) (Shao et al., 2024) method. By leveraging on-policy sampling and a goal-aware LLM judge with access to the ground-truth user goal, GRPO enables the Clarifier to learn question-asking strategies that explicitly reduce uncertainty about latent user intent.

Extensive experiments across domains and agent backbones demonstrate the effectiveness of learning-based clarification. We analyze how clarification trigger frequency influences both task success and interaction length, revealing a clear trade-off between performance and interaction cost. Although our clarifier is trained solely on interaction trajectories collected with Qwen3-8B as the underlying agent, it generalizes well to a diverse set of unseen agent backbones, consistently improving task success. Taken together, learned clarification outperforms the no-clarification baseline by 1.5%–12.6% while increasing interaction cost by only 0.1 steps on average, highlighting the value of explicitly trained clarification policies for robust and reliable tool-using agents.

2 Related Work

Clarification Questions and Proactive Question Asking. Prior work has studied clarification questions as a means of resolving ambiguity and missing information in dialogue and information-seeking systems (Clark, 1996; Traum and Larsson, 2003; Aliannejadi et al., 2019; Zou et al., 2023; Xia et al., 2025; Li et al., 2024a). Recent studies further examine large language models' ability to ask informative questions under incomplete information, often through dedicated benchmarks or diagnostic evaluations, revealing that strong language modeling does not necessarily yield effective proactive reasoning (Zhou et al., 2025). Complementary approaches frame question asking as a principled information gathering problem, for example by selecting queries that maximize expected information gain (Choudhury et al., 2025). However, these works primarily focus on benchmarking question-asking ability or identifying informative clarification questions, rather than learning clarification policies that can be reliably deployed within tool-using agents.

Tool-Using Agents. Recent work has studied large language models as tool-using agents capable of multi-step reasoning and decision making, enabled by integrating planning, tool invocation, and feedback-driven execution (Yao et al., 2022; Schick et al., 2023; Wölflein et al., 2025; Shen et al., 2024; Yuan et al., 2025; Zhu et al., 2025; Wu et al., 2024b; Lu et al., 2025; Wu et al., 2025; Liu et al., 2023b; Wang et al., 2024b). A growing body of benchmarks and environments further evaluate agent performance in complex, tool-rich settings (Qu et al., 2025; Liu et al., 2023a; Wang et al., 2024a; Yao et al., 2024; Andriushchenko et al., 2024; Xu et al., 2024; Chang et al., 2024; Li et al., 2024b; Wu et al., 2024a). A significant challenge, however, arises from the ambiguity inherent in real-world user instructions, which are often underspecified due to principles of linguistic economy or a user's limited domain knowledge. Existing frameworks, while proficient on well-defined tasks, falter in the face of such ambiguity. This leads to the agent's clarification behavior being tightly coupled with its operational policy, hindering independent control and optimization. To address this specific challenge, our work is situated within the τ-Bench environment (Yao et al., 2024). Its sophisticated simulation of user-LLM interactions and dynamic, multi-turn conversations makes it an ideal framework for studying how agents can effectively clarify ambiguous instructions.

3 Methodology

Our approach integrates the interaction environment built upon τ-Bench (Section 3.1), the collection of supervision, clarification, and data used for SFT, DPO, and GRPO training (Section 3.2), and the training setup for GRPO (Section 3.3). Together, these components enable the model to learn question-asking behaviors that minimize uncertainty about the underlying user intent.

3.1 Clarifier Interaction Environment

We build on τ-Bench, a benchmark for Tool–Agent–User interaction in real-world domains, as it explicitly models interactive task execution under partially observable user goals. Each task is formulated as a partially observable Markov decision process (POMDP) $(\mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, U)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $\mathcal{O}$ the observation space, T the transition function, R the reward function, and U the instruction specifying the user's task goal. The agent interacts with (i) domain-specific tool APIs and (ii) an LM-simulated user to complete U. We denote by $\mathcal{A}_{\text{tool}} \subseteq \mathcal{A}$ the subset of actions that invoke tools.

To model clarification behavior, we extend this interaction loop with a Clarifier module that is eligible to run only after tool invocations. When the agent issues a tool call $a_t \in \mathcal{A}_{\text{tool}}$ and receives the corresponding observation o_t, the Clarifier assesses whether the environmental feedback reveals latent ambiguity or missing task-critical information. This execution-then-clarify paradigm allows the agent to ground its questions in actual state constraints (e.g., search results) rather than speculative intent analysis. If clarification is needed, it produces a follow-up question added to a set of clarification turns $\mathcal{C}$, and the simulated user returns a corresponding response added to a set $\mathcal{R}$; otherwise no clarification turn is generated. These optional exchanges do not modify the underlying POMDP dynamics but enrich the recorded interaction trace.

Each multi-turn interaction produces a trajectory

$$\tau = \{\, U,\ (a_t, o_t)_{t=1}^{T},\ \mathcal{C},\ \mathcal{R},\ G \,\},$$

where (a_t, o_t) denotes the sequence of agent actions and tool observations, $\mathcal{C}$ and $\mathcal{R}$ collect the clarification questions and user responses produced during the episode, and G is the task-defined ground-truth user goal. All interaction data used to train our clarification model are collected from this Clarifier-augmented environment.

Figure 1 (a) depicts the core framework: the agent follows *Domain-Policy Tool Calling* constraints when interacting with tools, while the LM-simulated user follows a *User-Driven Fuzzy Intent* prompt. The Clarifier is triggered only after tool calls to assess whether a concise follow-up question is needed. Figure 1 (b) shows a representative τ-airline trajectory in which the Clarifier detects underspecified user intent and issues a targeted clarification that enables correct downstream tool use. A case study is provided in 7.5.

3.2 Training Data Construction

We construct three complementary datasets supporting SFT, pairwise preference learning via DPO, and on-policy optimization via GRPO. All data are collected from the Clarifier-augmented interaction environment described in Section 3.1, using trajectories generated from the 500 users in the τ-retail domain. The Clarifier agent is implemented using Qwen3-8B, while an LM-simulated user and a Qwen3-14B teacher model are employed to generate responses and clarification candidates. This choice is motivated by the results in Table 1, where Qwen3-14B provides consistent performance gains while maintaining a relatively small distributional gap between the teacher and the Qwen3-1.7B-Base student model. Sampling templates, detailed hyperparameter settings, and dataset format are provided in Appendix 7.1.

SFT. We collect 3,935 clarification-labeled examples from multi-turn interactions. Each instance includes a chain-of-thought (think-mode) output from the Qwen3-14B teacher model, which first reasons about the need for clarification and, if required, generates a final question. The dataset comprises 1,482 NO NEED TO ASK examples and 2,430 QUESTION examples, enabling the model to learn both when clarification is necessary and how to generate concise, task-relevant questions.

DPO. The DPO dataset consists of 1,859 pairwise comparisons. Each comparison includes two think-mode clarification candidates generated by the Qwen3-14B teacher model. A judge evaluates each pair based solely on the extracted decision (NO NEED TO ASK) or the final question, using the ground-truth user goal as reference. The judge assigns clear *win/lose* preferences to approximately 91% of the comparisons, while the remaining cases include 94 *tie* and 89 *both-bad* judgments. During training, we retain only comparisons with unambiguous *win/lose* outcomes and discard tie and both-bad cases, ensuring a clean and preference-consistent supervision signal.

GRPO. To curate the dataset, we employ a pairwise sampling and annotation protocol as a screening mechanism for discriminative difficulty. For each interaction state, we sample two offline candidate trajectories and evaluate them against the ground truth. We retain 3,141 instances where this comparison yielded distinct *win/lose* preferences or *both-bad* judgments, while discarding uninformative *tie* cases. Crucially, these offline candidates serve solely to identify challenging contexts and are explicitly discarded. We isolate only the dialogue history (x) and the task-specific ground-truth goal ($G^\star$). This ground truth provides the essential adjudication criteria for the goal-aware judge, allowing GRPO to refine the policy exclusively through on-policy rollouts focused on high-ambiguity contexts.

3.3 User-Goal-Driven GRPO for Questioning

While DPO supervises pairwise preferences, it does not explicitly model how a clarification question should be selected to best reveal the user's underlying goal. As illustrated in Figure 2, we instead formulate clarification as a problem of *goal-conditioned question selection*: given an interaction context x and the user's true goal $G^\star$, the optimal question is the one that most reduces uncertainty. In practice, this is implemented by sampling multiple candidate questions, evaluating them with a goal-aware judge, and optimizing the policy toward questions receiving higher goal-conditioned prefer-

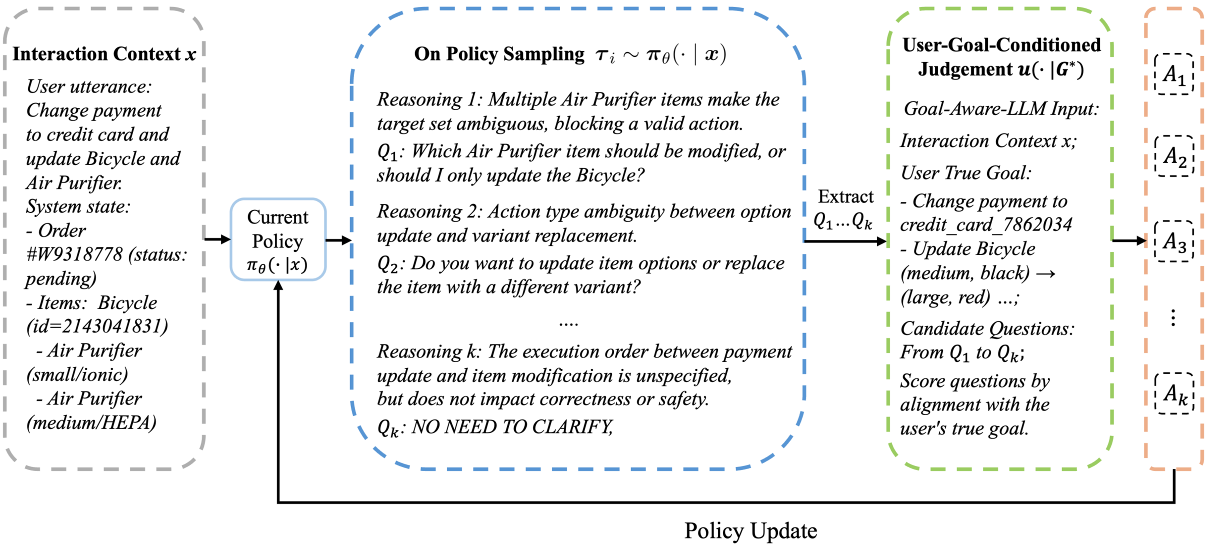


Figure 2: The model performs on-policy sampling to produce a set of candidate questions, which are then evaluated by a goal-aware LLM judge that returns utilities and an induced ranking. The resulting group-wise advantages are fed back into GRPO to refine the Clarifier's policy, enabling it to ask more goal-informative clarification questions.

ence signals. This formulation is related to active learning, but models uncertainty implicitly through preferences rather than explicit belief updates.

We apply GRPO with on-policy sampling, without rollout reuse. For each dialogue context x, which includes the dialogue history, the current user utterance, and the known system state, the current policy generates a group of k stochastic think-mode trajectories (sampled with temperature $T = 0.8$),

$$\tau_i \sim \pi_\theta(\cdot \mid x), \qquad i = 1, \dots, k,$$

where each trajectory τ_i contains an internal reasoning trace followed by a final output. From each trajectory, we extract only the final decision (NO NEED TO ASK) or clarification question, yielding a set of candidate clarifications

$$\mathcal{Q}(x) = \{Q_1, \dots, Q_k\}.$$

For evaluation, a goal-conditioned LLM judge with access to $G^\star$ scores the extracted outputs with scalar utilities $\{s_i\}_{i=1}^k$, assessing goal-oriented utility relative to $G^\star$ by measuring the information gain of questions or the judgment validity of non-clarification decisions. These scores induce a ranking

$$\mathrm{rank}\big(\mathcal{Q}(x) \mid G^\star\big).$$

The resulting scores define a goal-conditioned preference relation

$$Q_i \succ Q_j \iff u(Q_i \mid G^\star) > u(Q_j \mid G^\star),$$

where $u(\cdot \mid G^\star)$ denotes the judge's implicit goal-aligned utility function.

Following the group-wise preference signals obtained above, we convert the rankings or utilities into group-wise rewards $\{r_i\}_{i=1}^k$ and compute centered advantages:

$$A_i = r_i(G^\star) - \bar{r}, \qquad \bar{r} = \frac{1}{k}\sum_{j=1}^{k} r_j(G^\star).$$

Given k on-policy rollouts $\{\tau_i\}_{i=1}^k \sim \pi_\theta(\cdot \mid x)$, GRPO optimizes a KL-regularized policy gradient objective that encourages the Clarifier to produce clarification questions with higher goal-conditioned preference signals:

$$\begin{aligned}\mathcal{L}_{\mathrm{GRPO}}(\theta) = & -\mathbb{E}_{\tau\sim\pi_\theta(\cdot\mid x)}\big[A(\tau \mid G^\star)\, \log \pi_\theta(\tau \mid x)\big] \\ & + \beta\, \mathrm{KL}(\pi_\theta(\cdot \mid x) \,\|\, \pi_{\mathrm{ref}}(\cdot \mid x)).\end{aligned}$$

GRPO thus implements a fully on-policy optimization loop: the Clarifier samples multiple think-mode candidate questions, a goal-conditioned judge evaluates and ranks their extracted final outputs according to alignment with the user's true goal $G^\star$, and the resulting group-wise advantages guide policy updates. This training procedure enables the Clarifier to learn question-asking strategies that explicitly reduce uncertainty about the user's underlying intent.

4 Experiments

Building upon the framework introduced in Section 3, we train Clarifier models under the SFT,

Setting		Success Rate (%↑)			Steps ↓		
Training Strategy	**Based Model**	**Retail**	**Airline**	**Average**	**Average Agent**	**Average Clarify**	**Average Sum**
Baseline	Qwen3-1.7B-Base	16.5	13.3	14.9	30.6	1.00	31.6
SFT	Qwen3-1.7B-Base	**20.6**	15.3	18.0	30.6	3.23	33.83
DPO	Qwen3-1.7B-Base	17.7	14.0	15.9	31.3	2.87	34.17
DPO	SFT	18.3	11.3	14.8	31.5	2.96	34.46
GRPO (w/o $G^\star$)	Qwen3-1.7B-Base	17.1	14.6	15.9	29.3	3.24	32.54
GRPO	Qwen3-1.7B-Base	20.0	**16.7**	**18.4**	31.4	2.99	34.39
GRPO (w/o $G^\star$)	SFT	18.2	16	17.1	28.9	3.12	32.02
GRPO	SFT	19.4	15.3	17.4	30.2	2.92	33.12

Table 2: Pass@1 results on τ-Bench (`retail` and `airline`) using deterministic decoding (temperature = 0), averaged over three runs. *Average* implies the mean across domains. *Average Agent* refers to the mean number of interaction steps between the agent and user *excluding Clarifier turns*; *Average Clarify* counts average Clarifier invocations per task; and *Average Sum* is the total number of steps including Clarifier interactions. Methods annotated with (w/o $G^\star$) represent variants trained without access to ground truth information. The up (↑) and down (↓) arrows indicate that higher is better and lower is better, respectively.

DPO, and GRPO settings using the constructed datasets and evaluate them within the Clarifier-augmented τ-Bench environment. All models are fine-tuned *exclusively* on the 500 τ-retail training trajectories, and tested on 115 held-out `retail` tasks and 50 `airline` tasks, ensuring a clean separation between training and evaluation. The `airline` domain constitutes a fully out-of-distribution setting, featuring task structures, tool APIs, and interaction patterns that differ markedly from those observed during training. Implementation details and specific training hyperparameters can be found in Appendix 7.4.

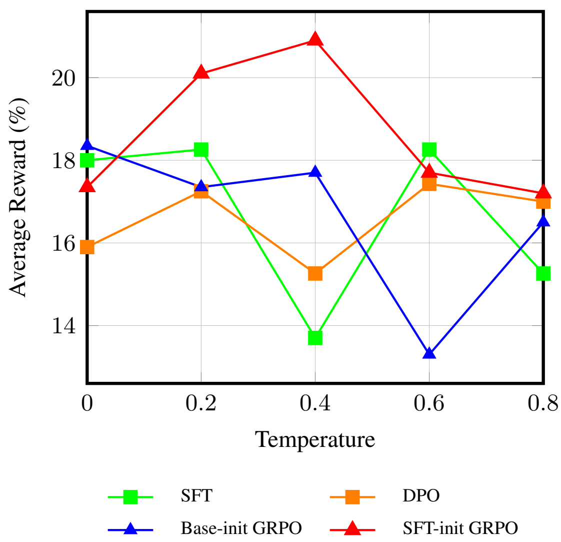
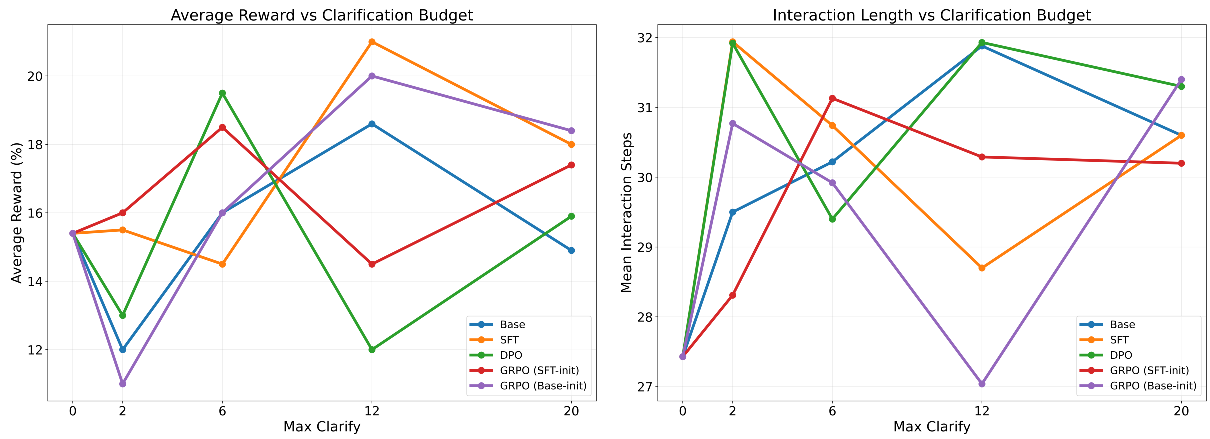


Figure 3: Effect of sampling temperature on clarification performance across different training strategies. Average Reward is computed as the average of Retail and Airline pass@1 accuracy.

4.1 Performance of Clarifier Interaction

Table 2 summarizes the performance of different Clarifier training strategies under deterministic decoding. Relative to the baseline, all fine-tuned Clarifier models substantially improve task success rate, indicating that learning to ask clarification questions meaningfully enhances downstream tool-use reliability. SFT attains the strongest in-domain accuracy on `retail` (20.6%), reflecting the effectiveness of supervised signals in shaping domain-aligned clarification behavior. While DPO shows mixed results: it improves the base model but degrades the SFT-initialized variant, indicating that DPO may conflict with SFT-induced behaviors.

GRPO delivers the most robust cross-domain performance. When initialized from the base model, GRPO achieves both the highest out-of-domain accuracy on `airline` (16.7%) and the best overall average (18.4%). This suggests that GRPO enables the Clarifier to acquire question-asking strategies that generalize beyond the training domain by more effectively probing latent user goals. However, this capability is contingent on ground truth access; training without such information (w/o G^*) causes the average accuracy to drop significantly to 15.9%, underscoring the necessity of ground-truth-derived rewards in guiding the optimization. Figure 3 examines how sampling temperature influences inference-time behavior. The GRPO model, particularly when initialized from the SFT checkpoint, achieves substantial gains at moderate temperatures (0.2–0.4), yielding noticeably higher mean Pass@1 compared with strict greedy decoding. In contrast, purely supervised or DPO-trained models do not benefit from higher



(a) Average Reward is computed as the domain-weighted average of Retail and Airline pass@1 accuracy.

(b) Mean number of interaction steps excluding clarification turns, averaged over the airline and retail domains.

Figure 4: Effect of clarification budget on task success and interaction efficiency under forced sampling.

sampling temperatures, and in some cases exhibit degraded performance. This indicates that GRPO learns a broader distribution of effective strategies, the benefits of which are maximized when moderate stochastic sampling enables the model to navigate a wider space of solution trajectories.

4.2 Impact of Clarification Trigger Frequency

We study the effect of clarification trigger frequency by varying the maximum number of allowed clarification turns under forced sampling. Figure 4 summarizes the resulting trade-off between task success and interaction efficiency.

As shown in Figure 4(a), increasing the clarification budget can improve Mean Pass@1, averaged over the airline and retail domains. However, the effect is non-monotonic across methods, with moderate clarification budgets consistently outperforming both no clarification and overly frequent clarification, suggesting that allowing more clarification opportunities does not uniformly translate into higher task success. Contrary to the intuition that more information is always beneficial, our findings indicate that over-clarification can actively degrade performance by introducing redundancy, polluting the task state, and increasing user fatigue. Figure 4(b) reports the corresponding interaction rounds, measured by the mean number of non-clarification steps. As the clarification budget increases, interaction trajectories generally become longer, reflecting the additional coordination and decision-making required during task execution, highlighting the inherent trade-off between task success and interaction efficiency.

Overall, these results indicate that clarification trigger frequency is a sensitive control parameter. While clarification can help resolve ambiguity introduced by stochastic user responses, overly frequent clarification may incur additional interaction cost without consistently improving task success.

4.3 Cross-Agent Generalization Ability

Table 3 presents the cross-agent generalization performance under deterministic decoding with different clarification strategies. All experiments are conducted with *think mode disabled* for all agents. We compare three clarification settings across agent backbones: no clarification (*None*), an external clarification model (*Qwen3-8B*), and our learned clarifier (*Ours*), trained via SFT followed by GRPO.

Across agents, incorporating clarification improves the average success rate by approximately **1.5–12.6%** compared to the no-clarification baseline, confirming the effectiveness of explicitly resolving fuzzy user intents for cross-agent generalization. While clarification increases steps for Qwen3-8B and Kimi-K2-Instruct, it reduces overall interaction length for other agents. Notably, when averaged across agents, *Ours* achieves a **5.2%** improvement in success rate while increasing the total number of interaction steps by only **0.1** on average, suggesting a favorable balance between improved task success and interaction efficiency. We further compare the learned clarifier (*Ours*) with an external pretrained clarifier (*Qwen3-8B*). At the agent level, the learned clarifier achieves higher success rates for several backbones, including Qwen3-8B, DeepSeek-R1, DeepSeek-V3.1, DeepSeek-V3.2, and Kimi-K2-Instruct, with improvements exceeding **5%**. In contrast, for Qwen3-

Setting		Success Rate (%↑)			Steps ↓		
Agent Model	**Clarifier Model**	**Retail**	**Airline**	**Average**	**Average Agent**	**Average Clarify**	**Average Sum**
Qwen3-8B	None	16.5	14.3	15.4	25.43	0	25.43
Qwen3-8B	Qwen3-8B	15.7	8.0	11.9	30.6	2.71	33.3
Qwen3-8B	Ours	**19.4**	**15.3**	**17.4**	30.2	2.92	33.12
Qwen3-14B	None	17.4	10	13.7	32.9	0	32.9
Qwen3-14B	Qwen3-8B	**24.4**	**28**	**26.4**	13.7	3.37	27.1
Qwen3-14B	Ours	24	20	22	21.8	3.12	24.9
Qwen3-30B-A3B	None	**17.4**	18	17.7	35.6	0	35.6
Qwen3-30B-A3B	Qwen3-8B	16.5	**22**	19.3	31	2.78	33.8
Qwen3-30B-A3B	Ours	**17.4**	**22**	**19.7**	32	3.0	35
DeepSeek-R1	None	31.3	40	35.7	26.1	0	26.1
DeepSeek-R1	Qwen3-8B	26.0	42	34	11.4	0.98	12.5
DeepSeek-R1	Ours	**33.9**	**44**	**39**	23.4	2.14	25.5
DeepSeek-V3.1	None	33.9	32	33	32.2	0	32.2
DeepSeek-V3.1	Qwen3-8B	40.9	38	39.5	25.8	2.67	28.5
DeepSeek-V3.1	Ours	**45.2**	**46**	**45.6**	28.2	2.66	30.9
DeepSeek-V3.2	None	36.5	30	33.3	32.9	0	32.9
DeepSeek-V3.2	Qwen3-8B	**39.1**	40	39.6	26	2.7	28.7
DeepSeek-V3.2	Ours	38.2	**42**	**40.1**	27.7	3	30.7
Kimi-K2-Instruct	None	13.0	32	22.5	28.8	0	28.8
Kimi-K2-Instruct	Qwen3-8B	6.1	30	18.1	29.7	0.72	30.4
Kimi-K2-Instruct	Ours	**13.9**	**34**	**24**	29.8	3.42	33.2
Agent Avg.	None	23.7	25.2	24.5	30.6	0.00	30.6
Agent Avg.	Qwen3-8B	24.1	29.7	27	24	2.3	27.8
Agent Avg.	Ours	**27.4**	**31.9**	**29.7**	27.6	2.89	30.5

Table 3: Agent and clarifier generalization performance with the clarifier decoded deterministically. Different agent backbones are compared under varying clarification strategies, including no clarification (*None*), an external clarification model (*Qwen3-8B*), and a learned clarifier (*Ours*) trained with SFT and GRPO. Bold numbers denote the best-performing configuration for each agent. ↑ (↓) indicates higher (lower) is better.

14B, the external clarifier attains a higher average success rate, exceeding that of *Ours* by 4.4%.

From an interaction perspective, the learned clarifier maintains a relatively consistent level of clarification across agent backbones, with the average number of clarification turns remaining within 2.1–3.2. This reflects a stable clarification strategy that does not aggressively minimize interaction steps. By contrast, when Qwen3-8B is used as the clarifier, a different interaction pattern is observed for certain agent backbones. In particular, for DeepSeek-R1 and Kimi-K2-Instruct, Qwen3-8B reduces the average number of clarification turns to below one (0.98 and 0.72, respectively), resulting in shorter interaction trajectories. However, this reduction in interaction length does not lead to higher task success, suggesting that strategies emphasizing fewer turns can shorten trajectories without reliably improving outcomes.

When averaged across agent backbones, *Ours* achieves a higher average success rate (**29.7%** vs. **27%**), accompanied by a modest increase in the number of clarification turns (2.89 vs. 2.3 on average), which leads to a slightly higher total interaction cost (30.5 vs. 27.8 steps). Despite its smaller model size (**1.7B** parameters), the learned clarifier achieves comparable or improved performance in the majority of settings, indicating that task-trained clarification can generalize across different agent backbones while requiring substantially less memory and inference cost.

5 Conclusion

We present a Clarifier-augmented framework based on τ-Bench and compare clarification learning via SFT, DPO, and GRPO. Learning to ask clarification questions consistently improves task success by 1.5–12.6% across domains and agent backbones. Further experiments on clarification budgets and cross-agent generalization show that the learned clarifier improves agent-averaged success by 5.2% while increasing interaction steps by only 0.1 on average, and generalizes effectively despite its smaller model size of 1.7B parameters.

6 Limitations

Our training data are constructed using a relatively coarse interaction-driven procedure and do not explicitly optimize for fine-grained coverage of all possible ambiguity patterns. Nevertheless, the proposed framework consistently achieves strong performance across domains, suggesting robustness to such data granularity. Exploring more refined data curation strategies remains an interesting direction for future work.

References

Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W Bruce Croft. 2019. Asking clarifying questions in open-domain information-seeking conversations. In *Proceedings of the 42nd international acm sigir conference on research and development in information retrieval*, pages 475–484.

Maksym Andriushchenko, Alexandra Souly, Mateusz Dziemian, Derek Duenas, Maxwell Lin, Justin Wang, Dan Hendrycks, Andy Zou, Zico Kolter, Matt Fredrikson, and 1 others. 2024. Agentharm: A benchmark for measuring harmfulness of llm agents. *arXiv preprint arXiv:2410.09024*.

Ma Chang, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. 2024. Agentboard: An analytical evaluation board of multi-turn llm agents. *Advances in neural information processing systems*, 37:74325–74362.

Deepro Choudhury, Sinead Williamson, Adam Goliński, Ning Miao, Freddie Bickford Smith, Michael Kirchhof, Yizhe Zhang, and Tom Rainforth. 2025. Bed-llm: Intelligent information gathering with llms and bayesian experimental design. *arXiv preprint arXiv:2508.21184*.

Herbert H Clark. 1996. *Using language*. Cambridge university press.

Bin Fu, Yunqi Qiu, Chengguang Tang, Yang Li, Haiyang Yu, and Jian Sun. 2020. A survey on complex question answering over knowledge base: Recent advances and challenges. *arXiv preprint arXiv:2007.13069*.

Dongyuan Li, Ying Zhang, Zhen Wang, Shiyin Tan, Satoshi Kosugi, and Manabu Okumura. 2024a. Active learning for abstractive text summarization via llm-determined curriculum and certainty gain maximization. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 8959–8971.

Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Erran Li Li, Ruohan Zhang, and 1 others. 2024b. Embodied agent interface: Benchmarking llms for embodied decision making. *Advances in Neural Information Processing Systems*, 37:100428–100534.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, and 1 others. 2023a. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. 2023b. Dynamic llm-agent network: An llm-agent collaboration framework with agent team optimization. *arXiv preprint arXiv:2310.02170*.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, and 1 others. 2025. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for llm tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1160–1183.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Yunjia Qi, Hao Peng, Xiaozhi Wang, Amy Xin, Youfeng Liu, Bin Xu, Lei Hou, and Juanzi Li. 2025. Agentif: Benchmarking instruction following of large language models in agentic scenarios. *arXiv preprint arXiv:2505.16944*.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. 2025. Tool learning with large language models: A survey. *Frontiers of Computer Science*, 19(8):198343.

Rafael Rafailov, Eric Zhang, Rose Liu, and Tatsunori Hashimoto. 2023. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*.

Sudha Rao and Hal Daumé III. 2018. Learning to ask good questions: Ranking clarification questions using neural expected value of perfect information. *arXiv preprint arXiv:1805.04655*.

T. Schick and 1 others. 2023. Toolformer: Language models can teach themselves to use tools. In *ICLR*.

Thomas Scialom, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano. 2019. Answers unite! unsupervised metrics for reinforced summarization models. *arXiv preprint arXiv:1909.01610*.

Zhihong Shao, Zihan Liu, Wei Zhang, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Weizhou Shen, Chenliang Li, Hongzhan Chen, Ming Yan, Xiaojun Quan, Hehong Chen, Ji Zhang, and Fei Huang. 2024. Small llms are weak tool learners: A multi-llm agent. *arXiv preprint arXiv:2401.07324*.

Manan Suri, Puneet Mathur, Nedim Lipka, Franck Dernoncourt, Ryan A Rossi, and Dinesh Manocha. 2025. Structured uncertainty guided clarification for llm agents. *arXiv preprint arXiv:2511.08798*.

David R Traum and Staffan Larsson. 2003. The information state approach to dialogue management. In *Current and new directions in discourse and dialogue*, pages 325–353. Springer.

Jize Wang, Ma Zerun, Yining Li, Songyang Zhang, Cailian Chen, Kai Chen, and Xinyi Le. 2024a. Gta: a benchmark for general tool agents. *Advances in Neural Information Processing Systems*, 37:75749–75790.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024b. Executable code actions elicit better llm agents. In *Forty-first International Conference on Machine Learning*.

Georg Wölflein, Dyke Ferber, Daniel Truhn, Ognjen Arandjelovic, and Jakob Nikolas Kather. 2025. Llm agents making agent tools. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 26092–26130.

Cheng-Kuang Wu, Zhi R Tam, Chieh-Yen Lin, Yun-Nung Chen, and Hung-yi Lee. 2024a. Streambench: Towards benchmarking continuous improvement of language agents. *Advances in Neural Information Processing Systems*, 37:107039–107063.

Junde Wu, Jiayuan Zhu, Yuyuan Liu, Min Xu, and Yueming Jin. 2025. Agentic reasoning: A streamlined framework for enhancing llm reasoning with agentic tools. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28489–28503.

Shirley Wu, Shiyu Zhao, Qian Huang, Kexin Huang, Michihiro Yasunaga, Kaidi Cao, Vassilis Ioannidis, Karthik Subbian, Jure Leskovec, and James Y Zou. 2024b. Avatar: Optimizing llm agents for tool usage via contrastive reasoning. *Advances in Neural Information Processing Systems*, 37:25981–26010.

Yu Xia, Subhojyoti Mukherjee, Zhouhang Xie, Junda Wu, Xintong Li, Ryan Aponte, Hanjia Lyu, Joe Barrow, Hongjie Chen, Franck Dernoncourt, and 1 others. 2025. From selection to generation: A survey of llm-based active learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14552–14569.

Frank F Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Z Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, and 1 others. 2024. Theagentcompany: benchmarking llm agents on consequential real world tasks. *arXiv preprint arXiv:2412.14161*.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. 2024. τ-bench: A benchmark for tool-agent-user interaction in real-world domains, 2024. *URL https://arxiv. org/abs/2406.12045*.

Shunyu Yao, Run-Ze Yang, Nan Cui, Karthik Narasimhan, and 1 others. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Asaf Yehudai, Lilach Eden, Alan Li, Guy Uziel, Yilun Zhao, Roy Bar-Haim, Arman Cohan, and Michal Shmueli-Scheuer. 2025. Survey on evaluation of llm-based agents. *arXiv preprint arXiv:2503.16416*.

Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Yongliang Shen, Kan Ren, Dongsheng Li, and Deqing Yang. 2025. Easytool: Enhancing llm-based agents with concise tool instruction. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 951–972.

Michael J.Q. Zhang, W. Bradley Knox, and Eunsol Choi. 2024. Modeling future conversation turns to teach llms to ask clarifying questions. *arXiv preprint arXiv:2410.13788*.

Zhanke Zhou, Xiao Feng, Zhaocheng Zhu, Jiangchao Yao, Sanmi Koyejo, and Bo Han. 2025. From passive to active reasoning: Can large language models ask the right questions under incomplete information? *arXiv preprint arXiv:2506.08295*.

Chenfei Zhu, Shao-Kang Hsia, Xiyun Hu, Ziyi Liu, Jingyu Shi, and Karthik Ramani. 2025. agentar: Creating augmented reality applications with tool-augmented llm-based autonomous agents. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*, pages 1–23.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

Jie Zou, Mohammad Aliannejadi, Evangelos Kanoulas, Maria Soledad Pera, and Yiqun Liu. 2023. Users meet clarifying questions: Toward a better understanding of user interactions for search clarification. *ACM Transactions on Information Systems*, 41(1):1–25.

7 Appendix

7.1 Prompt Templates and Generation Settings

We use the following decoding and sampling parameters for the *teacher model* throughout data generation:

```
"messages": messages,
"think_budget": 600,
"max_tokens": 1000,
"temperature": 0.6,
"top_p": 0.95,
"top_k": 30,
"stream": False,
```

Clarifier Prompt Template. The following prompt template defines the behavior of the clarification model, which conditions on the dialogue interaction history to determine whether a follow-up question is required and, if so, generates a concise clarification question.

```
"role_definition":
"You are a Clarification Assistant
responsible for determining whether the agent
needs to ask a clarifying question."
History: {history_text}
"instructions":
"Analyze the dialogue and decide whether
clarification is needed.
If not needed, answer ‘NO‘.
If needed, answer:
YES [QUESTION]Your clarification question
here[/QUESTION]
Ask only one concise question about user
intent.
Do not ask about data retrievable from tools.
Do not repeat previous questions."
"expected_output":
Either:
- NO
or
- YES [QUESTION]...[/QUESTION]
```

User Prompt Template. The following prompt template is used to simulate a human user during interaction, specifying the behavioral constraints and response style of the LM-based user simulator.

```
"role_definition":
"You are a user interacting with an agent.
Your behavior simulates a human user
following a hidden instruction."
"instructions":
"Generate only one message at a time.
Reveal information gradually instead of all
at once.
Do not invent facts missing from the
instruction—if the agent asks for unavailable
details, say you don’t remember.
When the instruction’s goal is completed,
output STOP.
Do not repeat the instruction verbatim; use
natural conversational phrasing.
Maintain a natural, human-like conversation
style."
"expected_output":
A single user utterance per step.
```

DPO Judge Prompt Template. The following prompt template is used for the preference judge in the DPO stage. Given two candidate clarification questions, the judge evaluates their relative quality with respect to task relevance, clarity, and usefulness, and produces a pairwise preference label used for training.

```
"role_definition":
"You are an expert evaluator for
clarification questions in customer service
scenarios."
User Detailed Requirement: {user_requirement}
Conversation History: {conversation_history}
"instructions":
"Evaluate the following two candidate
clarification questions. Consider the
following factors:
1. Relevance to the task and missing
information.
2. Clarity and specificity.
3. Naturalness and politeness.
4. Likelihood to elicit useful information
from the user."
"Your task":
"1. Think carefully about the evaluation.
2. Select the better question based on the
factors above."
"expected_output":
"Output your judgment in exactly this format:
PREFERENCE: [A/B/TIE/BOTH_BAD]
REASON: [Brief explanation]"
"rules":
"1. If one question is clearly better, choose
A or B.
2. If both are equally good, choose TIE.
3. If both are poor quality, choose
BOTH_BAD."
```

GRPO Judge Prompt Template. The following prompt template is used for the preference judge during GRPO training. Given multiple candidate clarification questions generated from on-policy rollouts, the judge ranks them by quality and relevance with respect to the user requirement and dialogue context, and assigns scalar scores that serve as group-wise preference signals for optimization.

```
"role_definition":
"You are a professional question quality
evaluator."
User Requirement: {user_requirement}
Original Dialogue: {original_prompt}
"instructions":
"Rank the following {n} clarification
questions from BEST to WORST based on their
quality and relevance.
You must consider the user requirement and
original dialogue context carefully.
- Output MUST be in English.
- Rank all {n} questions from best (1) to
worst ({n}).
- Assign a score in [0, 10] to each question
(higher is better)."
"expected_output":
"Ranking": {Ranking_output}
"Scores": {Scores_output}
```

7.2 Clarifier Example

This example illustrates a clarification scenario in which the agent cannot proceed using available tools alone and must request additional user-provided information. Although the user has specified the payment method and provided identity information, the order ID required for executing the exchange remains unspecified. The Clarifier correctly identifies this missing decision-critical input and generates a single, targeted clarification question to resolve the remaining uncertainty.

Clarifier Prompt.

```
You are a Clarification Assistant. Your goal
is to help the agent decide whether it needs
to ASK THE USER a clarifying or confirming
question before proceeding with the next
action.
Context:
User: I need to exchange some items via
PayPal. Let me check the details. /no_think
Agent: To assist you with exchanging items
via PayPal, I'll need the following
information:
1. Your email address or full name and zip
code to authenticate your account. 2. The
order ID of the delivered order you want to
exchange. 3. The item IDs of the items you
want to exchange. 4. The new item IDs you
would like to exchange the items for (ensure
they are of the same product type). 5.
Confirmation that you want to use PayPal as
the payment method for any price difference.
Please provide these details so I can proceed
with the exchange.
User: My name is Mei Martin and my zip code
is 32124. /no_think
Agent: [TOOL_CALL:
find_user_id_by_name_zip(first_name=Mei,
last_name=Martin, zip=32124)]
Tool[find_user_id_by_name_zip]:
mei_martin_4260
Your task:
1. Review the dialogue and detect any genuine
uncertainty about the user's intent or
decision. 2. If the next step is clear or
solvable with the agent's own tools or data,
do NOT ask. 3. Only if user input is truly
required, write ONE concise question that
directly removes the uncertainty.
Response format (STRICT):
- If clarification is needed: YES
[QUESTION]Your concise and specific
clarifying question to the user
here[/QUESTION] - If no clarification is
needed: NO
Rules:
- Ask ONLY about what the user personally
knows or decides (intent, preference,
confirmation). - NEVER ask about facts or
data retrievable from tools or databases. -
Do NOT repeat, paraphrase, or re-confirm what
the agent and user already said. - Avoid
repeating or rephrasing any previously asked
clarification questions.
Output only the answer in the specified
format, with no explanations or extra text.
/think
```

Clarify Reasoning (Chain-of-Thought).

```
Okay, let's see. The user wants to exchange
items via PayPal. They provided their name
and zip code, which the agent used to find
their user ID. Now, the next steps would be
to get the order ID, item IDs, new item IDs,
and confirmation about using PayPal for the
price difference.
The user hasn't given the order ID yet. The
agent needs that to proceed. Since the user
hasn't provided it, the agent needs to ask
for it. The order ID is something the user
would know, so asking for it is appropriate.
The agent already has the user's ID, but
cannot retrieve the order ID without user
input. Therefore, clarification is required.
```

Assistant Output.

```
YES [QUESTION]What is the order ID of the
delivered order you want to
exchange?[/QUESTION]
```

7.3 DPO Example

This example illustrates a case where the user specifies preferences at the *attribute level*, but the agent cannot map them to an executable action without additional user-provided identifiers. The Clarifier correctly detects that a minimal follow-up question is required. We present an example of a preference pair used for Direct Preference Optimization (DPO), where the clarification model is trained to prefer a more goal-resolving clarification question over a less effective alternative under the same interaction context.

Interaction Context

```
System Instruction: You are a Clarification
Assistant. Your goal is to help the agent
decide whether it needs to ask a clarifying
or confirming question before proceeding with
the next action.

Dialogue Context:
Clarification: Are you also requesting to
exchange the Mechanical Keyboard listed in
your order (#W7293142) for a different model?
User-Clarification: Yes, I also want to
exchange the Mechanical Keyboard {switch type:
clicky, backlight: RGB, size: full size} for
a keyboard with {backlight: none, size: 80%}
using the same gift card. /no_think
Agent: [TOOL_CALL:
exchange_delivered_order_items
(item_ids=[6956751343, 9025753381],
new_item_ids=[6956751343, 9025753381])]
Tool Response: Error — new item 9025753381
not found or available.
Clarification: Could you confirm the new item
IDs for the Mechanical Keyboard exchange?
User-Clarification: The new Mechanical
Keyboard I want is {switch type: clicky,
backlight: none, size: 80%} and I will use
the same gift card 9909795 for both exchanges.
/no_think
Agent: [TOOL_CALL:
get_product_details(product_id=1656367028)]
Tool Response: Returned available variants,
none matching size 80%.
```

Preferred Clarification (Chosen)

```
Output:
YES [QUESTION] What is the specific item ID
for the new Mechanical Keyboard with {switch
type: clicky, backlight: none, size: 80%}?
[/QUESTION]
```

Dispreferred Clarification (Rejected)

```
Output:
YES [QUESTION] What is the item ID for
[/QUESTION]
```

7.4 Training Hyperparameters

7.4.1 Supervised Fine-Tuning (SFT).

All Clarifier models are initialized from `Qwen3-1.7B-Base` and trained using full-parameter supervised fine-tuning with DeepSpeed. Training is performed on two GPUs with `bfloat16` precision. We use a learning rate of 1×10^{-6} and train for 3 epochs. The per-device batch size is set to 1 for both training and evaluation. Gradient checkpointing is enabled to reduce memory usage, and the maximum sequence length is set to 4096 tokens. A linear learning rate warmup with a warmup ratio of 0.05 is applied.

7.4.2 Direct Preference Optimization (DPO).

For DPO training, we initialize the model from `Qwen3-1.7B-Base` and apply full-parameter optimization using pairwise preference data. Training is conducted on two GPUs with `bfloat16` precision. We train for 2 epochs with a learning rate of 1×10^{-6}. The per-device batch size is set to 1, with gradient accumulation over 4 steps. The maximum sequence length is 2048 tokens. A KL regularization coefficient β is set to 0.2, and a linear warmup with a warmup ratio of 0.05 is applied.

7.4.3 Group-wise Reinforcement Preference Optimization (GRPO).

GRPO training is performed using an on-policy optimization setup with separate rollout and judge servers. The rollout server generates stochastic clarification candidates using the current policy, while a goal-aware judge model evaluates and ranks the sampled outputs to produce group-wise preference signals. All GRPO experiments are conducted with local rollout and judge servers running on dedicated GPUs.

We initialize GRPO from two different starting points: the base model (`Qwen3-1.7B-Base`) and the SFT-trained model. For the base-initialized setting, the model is trained for 200 optimization steps, while for the SFT-initialized setting, GRPO training is extended to 1000 steps to allow further refinement of clarification strategies. This asymmetric training schedule reflects the stronger prior provided by SFT initialization.

During GRPO training, we sample a fixed number of on-policy rollouts per update with a non-zero sampling temperature to encourage exploration. Policy updates are performed using a KL-regularized objective with respect to a fixed reference policy, controlled by a coefficient β. Gradient accumulation is applied to stabilize optimization, and model checkpoints are periodically saved for evaluation. Unless otherwise specified, all other training hyperparameters follow the default GRPO configuration.

We use the following settings for GRPO unless otherwise specified:
(i) training epochs: 3;
(ii) rollout temperature: 0.8;
(iii) per-update batch size: 1;
(iv) learning rate: 5×10^{-7};
(v) gradient accumulation steps: 8;
(vi) KL coefficient: $\beta = 0.1$;
(vii) checkpointing every 100 update steps.

(viii) on-policy rollouts per update: $k = 8$;
We perform 200 update steps for *Base-init GRPO* and 1000 update steps for *SFT-init GRPO*.

7.5 Case Study: Clarification as a Critical Control Mechanism

7.5.1 Task Overview

We analyze a real-world tool-augmented conversational agent trajectory in a retail domain. The task requires modifying a pending order by changing both the payment method and a specific item variant. The agent operates under strict system constraints, including irreversible tool calls and limited action opportunities.

Reward Signal. The trajectory receives a terminal reward of 1.0 with full action credit ($r_{\text{actions}} = 1.0$), indicating successful task completion consistent with the ground-truth action sequence.

7.5.2 System Policy and Constraints

The agent follows a predefined retail policy with the following critical constraints:

- User identity must be authenticated before any action.
- Only one user can be served per conversation.
- Only orders with status `pending` may be modified.
- Payment method modification and item modification are separate tool calls.
- `modify_pending_order_items`:
  - Can be invoked **only once**.
  - Is irreversible.
  - Requires collecting all items to be modified in a single call.
  - Only allows replacement with an available item variant of the same product.

These constraints significantly restrict the agent's error tolerance.

7.5.3 Task Specification

The following task is synthetically annotated and corresponds to the trajectory analyzed in this work.

Task Definition.

Annotator: synthetic
User ID: `lucas_martin_4549`

User Instruction.

Your name is Lucas Martin and your email is lucas.martin5733@example.com. You are patient, cautious, and organized. For order #W9318778, change the payment method to `credit_card_7862034`. For order #W9318778, modify the Bicycle from {`frame size: medium, color: black, type: mountain`} to {`frame size: large, color: red`}; modify the Air Purifier from {`room size: medium, filter type: HEPA, features: quiet operation`} to {}. All modifications should be processed via `credit_card_7862034`.

Target Actions. The task specifies the following executable actions: **Modify Payment Method**

```
Action(
 name = "modify_pending_order_payment",
 kwargs = {
 order_id: "#W9318778",
 payment_method_id: "credit_card_7862034"
    }
)
```

Modify Order Items

```
Action(
 name = "modify_pending_order_items",
 kwargs = {
 order_id: "#W9318778",
 item_ids: ["2143041831", "3076708684"],
 new_item_ids: ["5606522780", "3076708684"],
 payment_method_id: "credit_card_7862034"
    }
)
```

7.5.4 Complete Interaction Trajectory

1. **User:** Requests to modify an order.
2. **Agent:** Requests identity verification.
3. **User:** Provides name and email.
4. **Tool Call:** `find_user_id_by_email`.
5. **Tool Result:** User ID retrieved successfully.

Order Retrieval

1. **User:** Provides order ID #W9318778.
2. **Tool Call:** `get_order_details`.

3. **Tool Result:**
   - Order status: `pending`
   - Items:
     - Bicycle (item ID 2143041831)
     - Mechanical Keyboard
     - Wall Clock
     - Air Purifier (item ID 5669664287)
     - Air Purifier (item ID 3076708684)
   - Original payment method: gift card

User Intent Declaration

1. **User:** Requests:
   - Change payment method to `credit_card_7862034`.
   - Update both Bicycle and Air Purifier items.

Payment Modification

1. **Tool Call:** `modify_pending_order_payment`.
2. **Tool Result:**
   - Payment method updated.
   - Original gift card payment refunded.
   - Order status remains `pending`.

Clarification Turn 1: Item Scope Conflict

1. **Agent (Clarify):** Requests clarification regarding which item should be updated, noting inconsistency between the user request and the system state.
2. **User:** Retracts Air Purifier modification and specifies updating the Bicycle to a large frame size and red color.

Failed Item Modification Attempt

1. **Tool Call:** `modify_pending_order_items` with a non-existent new item ID.
2. **Tool Result:** `Error: new item not found or available`.

This confirms that the agent cannot safely infer a valid item variant without additional information.

Clarification Turn 2: Action Type Alignment

1. **Agent (Clarify):** Asks whether the user intends to update item options or replace the item.
2. **User:** Confirms intent to update attributes (frame size and color).

This exposes a mismatch between natural language intent and system-supported operations.

Product Variant Discovery

1. **Tool Call:** `get_product_details`.
2. **Tool Result:** Multiple Bicycle variants are returned, including:
   - Variant 5606522780 (large, red, available).

Clarification Turn 3: Irreversible Action Confirmation

1. **Agent (Clarify):** Requests explicit confirmation of variant ID 5606522780 before proceeding.
2. **User:** Confirms the variant selection.

Final Item Modification

1. **Tool Call:** `modify_pending_order_items`.
2. **Tool Result:**
   - Bicycle item replaced successfully.
   - Order status updated to `pending (item modified)`.
   - Correct payment adjustment applied.

Causal Analysis of Clarification

Failure Risks Without Clarification

Without clarification, the agent would face unavoidable failure modes:

- Acting on incorrect item scope (Air Purifier vs. Bicycle).
- Attempting unsupported option-level modifications.
- Selecting invalid or unavailable item variants.
- Premature invocation of an irreversible API.

Any of these failures would result in a terminal reward of zero.

Clarification as a Necessary Condition

Each clarification turn resolves a distinct blocking issue:

- **Clarify 1:** Aligns target item with system state.

- **Clarify 2:** Maps user intent to legal action space.
- **Clarify 3:** Safeguards an irreversible tool call.

Notably, a tool-level failure occurs prior to sufficient clarification, demonstrating that success is unattainable without corrective clarification.

Conclusion

Clarification is not an auxiliary conversational feature in this task. It is a necessary execution-level control mechanism. Without clarification, no valid action sequence exists that satisfies system constraints. Thus, clarification is a necessary and causal factor for task success in this trajectory.

Clarification enables the existence of a successful trajectory under irreversible and highly constrained tool-use settings.