

# MAC: A Multi-Agent Framework for Interactive User Clarification in Multi-turn Conversations

Emre Can Acikgoz<sup>1</sup>, Jino Oh<sup>2</sup>, Joo Hyuk Jeon<sup>2</sup>, Jie Hao<sup>2</sup>,  
Heng Ji<sup>2</sup>, Dilek Hakkani-Tür<sup>2</sup>, Gokhan Tur<sup>2</sup>, Xiang Li<sup>2</sup>, Chengyuan Ma<sup>2</sup>, Xing Fan<sup>2</sup>

<sup>1</sup>University of Illinois Urbana-Champaign, <sup>2</sup>Amazon Alexa  
acikgoz2@illinois.edu, ojino@amazon.com

## Abstract

Conversational agents often encounter ambiguous user requests, requiring an effective clarification to successfully complete tasks. While recent advancements in real-world applications favor multi-agent architectures to manage complex conversational scenarios efficiently, ambiguity resolution remains a critical and underexplored challenge—particularly due to the difficulty of determining which agent should initiate a clarification and how agents should coordinate their actions when faced with uncertain or incomplete user input. The fundamental questions of when to interrupt a user and how to formulate the optimal clarification query within the most optimal multi-agent settings remain open. In this paper, we propose **MAC (Multi-Agent Clarification)**, an interactive multi-agent framework specifically optimized to resolve user ambiguities by strategically managing clarification dialogues. We first introduce a novel taxonomy categorizing user ambiguities to systematically guide clarification strategies. Then, we present MAC that autonomously coordinates multiple agents to interact synergistically with users. Empirical evaluations on MultiWOZ 2.4 demonstrate that enabling clarification at both levels increases task success rate 7.8% (54.5 → 62.3) and reduces the average number of dialogue turns (6.53 → 4.86) by eliciting all required user information up front and minimizing repetition. Our findings highlight the importance of active user interaction and role-aware clarification for more reliable human-agent communication.

## 1 Introduction

Effective user clarification is fundamental to conversational agents, significantly impacting their ability to fulfill user requests accurately and efficiently (Aliannejadi et al., 2019). In natural interactions, users often express ambiguous queries, intentionally or unintentionally omitting details that seem inferable or contextually obvious. Such am-

**MultiWOZ 2.4**

**User:** Hey! I need to book a restaurant for this Friday at 8pm.

**Assistant (Manager):**  
**Think:** The user's query is missing the group size, which is necessary for any booking. Cuisine details are domain-specific, so I will leave that to the expert.  
**Clarify:** How many people will be dining?

**User:** There will be 7 of us.

**Assistant (Manager):**  
**Think:** Now I know the group size, date, and time. The cuisine preference is still unknown, but it is a domain-specific detail and I will route it to Restaurant Agent.  
**Route:** Restaurant Agent

**Assistant (Restaurant Agent):**  
**Think:** The manager has collected the general details. Now, as the expert, I need to clarify the user's cuisine preference to proceed with the booking.  
**Clarify:** Do you have a cuisine preference or a particular restaurant?

**User:** We would like Italian food.

**Assistant (Restaurant Agent):**  
**Think:** I have the cuisine, but not the specific restaurant. I should clarify options.  
**Clarify:** Here are three Italian restaurants available at 8pm: Trattoria Bella, Roma's Kitchen, and Olive Garden. Which would you like to book?

**User:** Let's go with Trattoria Bella.

**Assistant (Restaurant Agent):**  
**Think:** All necessary information is now collected. I can proceed with reservation.  
**Response:** Your table for 7 at Trattoria Bella this Friday at 8pm is booked.

**Figure 1: Multi-Agent Dialogue Illustrating Harmonized Clarification.** The manager agent first identifies and resolves only high-level, domain-agnostic ambiguity (group size), explicitly leaving domain-specific clarifications (such as cuisine and restaurant selection) to the domain expert which can require domain knowledge from the database. This dialogue illustrates how each agent's role is confined to its designated scope: the manager collects general requirements, while the expert gathers and confirms specialized details before completing the reservation.

biguity can cause agents to make incorrect assumptions, provide incomplete responses, or even fail to accomplish tasks—issues that are especially critical in high-stakes domains such as healthcare, finance, and customer support. Proactively resolving ambiguities through targeted user interactions by asking clear and relevant clarification questions can substantially enhance the accuracy of task execution, user satisfaction, and the overall effectiveness of conversational systems (Deng et al., 2023; Acikgoz et al., 2025e).

In single-agent systems, the challenge of ambiguity resolution has been previously studied with different strategies (Dongre et al., 2024), from asking targeted questions (Li et al., 2023; Zhang and Choi, 2025) to inferring user preferences from past interactions (Andukuri et al., 2024). However, the

landscape of conversational AI is rapidly evolving towards more complex, multi-agent architectures, especially in industrial settings where a single agent cannot efficiently manage the large number of APIs and multitasking demands (Sun et al., 2025). As a result, manager–expert routing systems are becoming the standard for handling real-world tasks (Guo et al., 2024; Tran et al., 2025). This paradigm, often featuring a "manager" or "advisory" agent that routes requests to specialized "expert" agents, introduces new layers of complexity for user interaction (Ong et al., 2025). In such a setup, determining the optimal moment and method for clarification becomes a significant challenge. For instance, should the high-level advisory agent, which first receives the user’s request, interrupt for clarification, or should this be delegated to a domain-specific expert agent, potentially increasing latency and conversational turns? Moreover, deciding how much domain-specific knowledge the manager should possess introduces another design challenge, establishing an essential knowledge boundary. Therefore, proposing an approach that effectively manages ambiguity resolution while remaining independent of this specific design choice is crucial for creating flexible and robust multi-agent systems.

To explore these open questions, we introduce MAC (Multi-Agent Clarification), the first framework that focuses on resolving user ambiguities within multi-agent conversational systems and aims to uncover how, when, and by whom user clarifications should be initiated within these multi-agent settings (See Figure 1). The framework strategically determines not only the moment to seek clarification but also which agent—the supervisor or the domain-specific expert—is best positioned to ask. Our experiments on MultiWOZ 2.4 (Ye et al., 2022) reveal that the placement and timing of clarification matter more than previously recognized. First, we show that enabling clarification at both levels delivers not just higher task success with a 7.8% absolute gain over the no-clarification baseline, but does so while reducing the average number of conversational turns. Second, the optimal coordinated setup—where the supervisor manages high-level ambiguities and the expert agent resolves domain-specific ones—delivers the highest performance, even outperforming previous state-of-the-art TOD approaches on MultiWOZ with 11.50%. This means effective clarification is not merely about “asking more questions”, but about delegating the right questions to the right agents at

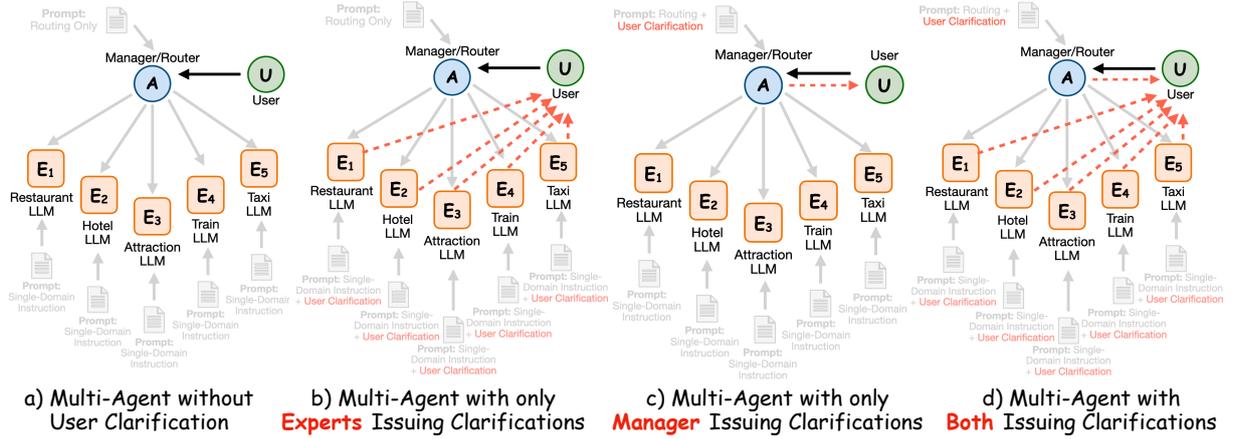
the right time.

The main contributions of this work are:

- We formalize the ambiguity resolution problem in **multi-agent conversational systems** with a taxonomy (Table 1), where decisions about *when*, *who*, and *how* to clarify are jointly optimized among different agents.
- We propose **MAC**, the first multi-agent framework for user clarification, enabling distributed agents to dynamically coordinate clarification behavior.
- We show that coordinated clarification, when both manager and experts are empowered to ask targeted questions, leads to a 7.8% absolute increase in task success (54.5% → 62.3%) while also reducing average dialogue length (6.53 → 4.86 turns) on MultiWOZ 2.4.
- We conduct extensive ablation studies, benchmarking MAC against strong single- and multi-agent baselines, analyzing the trade-offs of different clarification strategies, and demonstrating that our modular approach consistently outperforms prior TOD systems by a substantial margin. Additionally, we show that these gains are robust across diverse LLM backbones, including both proprietary and open-source models.

## 2 Related Work

**Asking Clarification Questions** Asking user clarification questions has been studied in conversational AI research, with distinct focuses on when to ask and what to ask (Kuhn et al., 2022). Some approaches use uncertainty estimation or information-theoretic models to decide when to initiate clarification (Zhang and Choi, 2023). More advanced frameworks attempt to address both when and what to ask problems jointly (Andukuri et al., 2024; Zhang et al., 2024), but they are often limited to a small number of conversational turns, which is insufficient for complex, real-world tasks. Closest to our approach, ReSpAct (Dongre et al., 2024) enables clarification with rule-based prompting, however operating in vanilla single-agent settings. This approach fails to address the challenges of production systems, such as smart home platforms, which operate as complex multi-agent systems (Guo et al., 2024). Our work fills this critical gap by proposing a novel framework for coordinating clarification strategies across multiple specialized agents to ensure a coherent and efficient user experience similar to direct real world settings.



**Figure 2: A comparison of multi-agent architectures for user clarification in task-oriented dialogue.** (a) **Baseline:** The manager/router agent routes user queries to domain-specific expert LLMs without any clarification. (b) **Experts-only clarification:** Each expert LLM independently interacts with the user to request clarification when needed, enhancing single-domain understanding. (c) **Manager-only clarification:** The manager/router agent requests clarification from the user before routing to any expert, enabling global disambiguation. (d) **Both:** Both the manager/router and the expert LLMs can independently interact with the user for clarification, allowing multi-level user-agent interaction. Dashed red arrows denote user clarification turns.

**Large Language Models for Task-Oriented Dialogue** Recent progress in LLMs has led to their adoption in multi-domain TOD systems (Hudeček and Dusek, 2023). Existing approaches typically rely on either prompting-based methods (Hu et al., 2022; Chung et al., 2023; Xu et al., 2024) or specialized fine-tuning (Hosseini-Asl et al., 2020; Yang et al., 2021; Zhong et al., 2023; Sun et al., 2023; Bang et al., 2023; Li et al., 2024). Fine-tuned models are often tailored to narrow tasks such as state tracking or offline benchmarks, and as a result, they struggle to generalize to complex, real-world, multi-turn conversations (Acikgoz et al., 2025a). More recently, AutoTOD (Xu et al., 2024) demonstrated the use of GPT-4 with domain-specific, hand-crafted prompts and pre-defined APIs, but this approach depends heavily on lengthy instructions and lacks adaptability. On the other hand, some recent studies have begun to explore multi-agent architectures for TOD (Gupta et al., 2024), but they have not addressed the crucial aspect of user clarification, which is an essential skill for handling ambiguous or incomplete user requests in practical settings. In contrast, our work introduces the first multi-agent system explicitly focusing on asking user clarification question in multi-turn TOD, establishing an optimal framework for more reliable and user-centric dialogue agents.

### 3 Environment

**MultiWOZ 2.4** In our research, we utilize MultiWOZ 2.4 (Ye et al., 2022), a comprehensive multi-domain dialogue benchmark designed for

task-oriented dialogue (TOD) systems. It contains multi-turn conversations between a user and a system simulating a dialogue assistant. The user is given a goal (e.g., book a hotel and a restaurant in the same area), while the system must fulfill the request using a consistent belief state and database API. Each dialogue is annotated with dialogue acts, belief states, and system actions at each turn, enabling full end-to-end modeling and evaluation. It consists of  $\sim 8,500$  training dialogues and a test set of 1,000 dialogues. The test conversations within MultiWOZ 2.4 simulate customer service interactions across five distinct domains: restaurant, hotel, train, attraction, and taxi. To create more realistic real-world scenarios in multi-turn settings with actual users, we enhanced this dataset by incorporating a user simulator, making the tasks both more challenging and more authentic.

**Task** During evaluation, each dialogue involves between up to five domains: *restaurant*, *hotel*, *train*, *attraction*, and *taxi*. The agent must understand the user’s multi-intent goals, track the evolving belief state, issue database queries, and generate appropriate system responses. Crucially, user queries often underspecify constraints (e.g., "Book a restaurant for dinner"), requiring the agent to proactively request clarification (e.g., number of people, cuisine, or time), which makes a well-suited environment to test our multi-agent approach.

**User Simulator** Our experimental setup involves a user simulator, which we implemented to interact with the agent in a multi-turn conversational flow.

Agent	Clarification Category	Description / Example
Supervisor	<ol style="list-style-type: none"> <li>1. Domain Ambiguity</li> <li>2. Intent Ambiguity</li> <li>3. Vague Goal Specification</li> <li>4. Contextual Disambiguation</li> <li>5. General Conflict</li> <li>6. General Noise/Correction</li> <li>7. Unfamiliar Domain Request</li> </ol>	<p>User query could match multiple domains (e.g., “<i>Find me a good place.</i>”)</p> <p>Domain is clear, but user’s goal is not (“<i>Tell me about trains.</i>”)</p> <p>Query is too broad to act on (“<i>Help me with my trip.</i>”)</p> <p>Vague referents like “it” or “that place” are unclear</p> <p>Broad contradiction in user input (“<i>I changed my mind about the date.</i>”)</p> <p>Common errors/typos needing clarification (“<i>I meant tomorrow not today.</i>”)</p> <p>Request does not match any known domain (“<i>Can you fix my phone?</i>”)</p>
Expert	<ol style="list-style-type: none"> <li>1. Parameter Underspecification</li> <li>2. Value Ambiguity/Vagueness</li> <li>3. Constraint Conflict</li> <li>4. Entity Disambiguation/Not Found</li> <li>5. Confirmation of Inferred Information</li> </ol>	<p>Missing key slot values (e.g., <i>location, cuisine, people, time</i>)</p> <p>Subjective terms require clarification (“<i>a nice place</i>”)</p> <p>Contradictory constraints (“<i>a cheap but expensive restaurant</i>”)</p> <p>Ambiguous or unrecognized entity (<i>restaurant name not found</i>)</p> <p>Inferred detail from context needs user confirmation</p>

**Table 1: Clarification Taxonomies for Ambiguity Handling: Supervisor and Experts.** Supervisor agent addresses only high-level, domain-agnostic ambiguities, while the Expert agent resolves domain-specific underspecification prior to API execution.

This simulator is tasked with pursuing the predefined user goals, while the agent’s objective is to assist the user in achieving these goals by interacting with an external database. We chose to work with MultiWOZ 2.4 over other benchmarks, due to its capability to simultaneously handle five distinct domains within a single conversational context. The available actions for each domain are detailed in Appendix Table 6.

## 4 Method

Ambiguity in user requests is a central challenge in conversational agents, yet there is limited empirical guidance in multi-agent dialogue systems on where and how clarification should be initiated in such frameworks. To address this, we systematically investigate different agent-level strategies for user clarification in a hierarchical multi-agent architecture comprising a manager/router and multiple domain-specific experts (See Figure 2).

### 4.1 MAC: Multi-Agent Clarification for User Ambiguities

In our multi-agent framework, we adopt a centralized multi-agent setting as our base architecture, which consists of a single **supervisor agent** and multiple specialized **domain expert agents**. The supervisor agent is responsible for orchestrating the overall dialogue flow by routing each user request to the most relevant domain expert, following the router-based approach in Ong et al. (2025). Each expert agent is specialized for one of the five domains in MultiWOZ 2.4, and is tasked with executing domain-specific actions to fulfill user goals. In MAC, we further enhance this framework by integrating user clarification mechanisms to re-

solve ambiguities. This involves assigning specific clarification-handling capabilities to both supervisor and domain expert roles as in Table 1, enabling them to manage different forms of uncertainty and improve final task outcomes.

**Supervisor Agent** In the MAC framework, the Supervisor agent is responsible for two different tasks: (i) orchestrating the agent collaboration by routing user queries to the appropriate domain expert, and (ii) handling top-level clarification of user requests when the ambiguity can be resolved with general commonsense reasoning, independent of domain-specific knowledge (Table 1, top). Formally, for each incoming user query  $u$ , the Supervisor evaluates an ambiguity function  $is\_ambiguous(u) \in 0, 1$ : if  $is\_ambiguous(u) = 1$ , the agent issues a clarification prompt to the user using the standardized format `<clarify>question</clarify>`; otherwise, it selects the appropriate domain expert with `<route>domain</route>`. This prompt-based control flow is illustrated in Figure 7, where the Supervisor’s output is parsed and dispatched to downstream agents. Notably, the Supervisor operates without access to domain-specific databases or APIs, ensuring that only non-domain-specific ambiguities (e.g., group size or intent) are addressed at this stage. After resolving high-level ambiguities, supervisor delegates the (potentially clarified) user request to the corresponding domain expert, enabling more efficient and role-aware collaboration across the agent hierarchy.

**Domain Expert Agents** Each Domain Expert agent is responsible for executing user goals within a specific task domain. We instantiate

---

**Algorithm 1** MAC: Multi-Agent Clarification Workflow

---

**Require:** User query  $u_t$  at dialogue turn  $t$ ; supervisor agent  $A_S$ ; domain experts  $A_E = \{A_{d_1}, \dots, A_{d_n}\}$

**Ensure:** Either CLARIFY( $q$ ) or RESPOND( $r$ )

```
1 function MAC( $u_t$ )
2   if  $A_S.is\_ambiguous(u_t)$  then
3      $q \leftarrow A_S.ask\_clarification(u_t)$ 
4     return CLARIFY( $q$ ) ▷ Supervisor requests disambiguation
5   end if
6    $d \leftarrow A_S.select\_domain(u_t)$ 
7    $A_d \leftarrow$  the expert for domain  $d$  ▷ Route to best-fit expert
8   if  $A_d.is\_ambiguous(u_t)$  then
9      $q_d \leftarrow A_d.ask\_clarification(u_t)$ 
10    return CLARIFY( $q_d$ ) ▷ Expert requests a targeted follow-up
11  else
12     $r \leftarrow A_d.execute\_domain\_response(u_t)$ 
13    return RESPOND( $r$ ) ▷ Final, domain-grounded answer
14  end if
15 end function
```

**Design notes.** (i) Only one clarification is issued per turn to limit latency. (ii)  $A_S$  handles global ambiguity;  $A_d$  handles domain-specific gaps. (iii) Routing uses  $A_S.select\_domain$ , which may rely on intent classification or retrieval over domain schemas.

---

five expert agents, corresponding to the five domains in MultiWOZ 2.4: restaurant, hotel, train, taxi, and attraction. Once a user query is routed to a domain expert, the agent analyzes the input—potentially enriched by prior supervisor-level clarification—and determines whether the information is sufficient to proceed with an *accurate API calls* or *reliable response generation*. To guide this behavior, we prompt each expert individually with domain-specific instructions that are coupled with the standardized protocols for user clarification (see Figure 8), following predefined expert specific clarification taxonomy (Table 1, bottom). Similar to the supervisor, the agent computes an ambiguity function  $is\_ambiguous(u) \in \{0, 1\}$ ; if the result is 1, the agent triggers a clarification request formatted as `<clarify>question</clarify>`. If the input is deemed sufficient, the agent executes the necessary domain-specific operations and responds using the structure `<response>utterance</response>`. Otherwise, the agent is allowed to ask multiple clarification questions until the total conversation length exceeds 20 turns. These prompt-structured outputs allow the framework to dynamically interleave reasoning, clarification, and execution in multi-turn interactions. Domain Experts have access to API schemas and databases corresponding to their domain, enabling them to ground their responses in task-specific constraints and complete user requests accurately.

To elucidate the core principles of MAC, we conduct a systematic analysis of different strategies

across the experimental design choices detailed in Figure 2.

## 5 Experiments and Results

### 5.1 Experimental Setup

In our MAC framework, we used gpt-4o-2024-08-06 as the base configuration for the selected LLM, serving as both the advisor and each expert, unless otherwise specified. However, we have conducted comprehensive ablation studies on the effect of model choice for both nodes in the Section 5.4. We conducted our evaluation on the MultiWOZ 2.4 test split, which contains 1,000 test samples from five domains: restaurant, hotel, train, attraction, and taxi. The evaluation was performed in online sessions where we implemented a user simulator based on gpt-4o-2024-08-06, as defined in Section 3. To account for LLM randomness, we ran each experiment five times and report the Success Rate with Avg@5 with their standard deviations and also include Success Rate with Max@5 which gives the max scores achieved in these five runs. Further details about the evaluation metrics can be seen in Section A.

### 5.2 Baselines

MAC is the first LLM-based multi-agent framework specifically designed for user clarification. To evaluate its effectiveness, we compare it against three variants of the same multi-agent architecture: (i) without any user clarification (see Figures 5 and 6 for baseline supervisor and expert prompts

Method	Clarification	Success (Max@5 $\uparrow$ )	Success (Avg@5 $\uparrow$ )	Avg. Turns ( $\downarrow$ )
MAC w/o Clarification	-	54.5	53.72 $\pm$ 0.92	6.53
MAC <sub>expert</sub>	Expert	55.6	54.88 $\pm$ 1.04	5.53
MAC <sub>supervisor</sub>	Supervisor	57.1	55.50 $\pm$ 1.86	5.11
MAC	Both	<b>62.3</b>	<b>58.40 <math>\pm</math> 2.10</b>	<b>4.86</b>

**Table 2: Main results on MultiWOZ 2.4.** Main results comparing different prompting and clarification strategies in the MAC framework on MultiWOZ 2.4. We report (**Success Max@5**): the highest single-run task success rate out of five runs, (**Success Avg@5**): the mean and standard deviation of success rates over five runs, and (**Avg. Turns**): the average number of dialogue turns per conversation (lower is better). Each row corresponds to a specific agent configuration—clarification enabled for the expert, the supervisor, both, or neither. Results demonstrate that enabling clarification for both supervisor and expert agents leads to the highest task success and most efficient dialogues.

without clarification), (ii) with user clarification handled only by the Supervisor, and (iii) with user clarification enabled only for the domain experts (see Figure 2). In setting (i), neither the Supervisor nor the domain experts are instructed to ask clarification questions. In setting (ii), only the Supervisor is prompted to perform both routing and user clarification, while the domain experts are limited to responding after routing. In setting (iii), only the domain experts are prompted to ask clarification questions, and the Supervisor is responsible solely for routing. In contrast, MAC enables user clarification at both the Supervisor and domain expert levels, allowing every agent node to interact with the user as needed. This setup allows for a fair and systematic evaluation of the individual and combined effects of clarification skills across different nodes.

### 5.3 Main Results

We compare MAC against three variants: (i) MAC without any clarification capability, (ii) MAC where only domain-specific experts perform clarifications (MAC<sub>expert</sub>), and (iii) MAC where only the supervisor at the top node initiates clarification questions (MAC<sub>supervisor</sub>). Table 2 summarizes our main findings, demonstrating the effectiveness of the proposed MAC framework. Specifically, our proposed MAC framework achieves an increase in task accuracy of approximately 8% at maximum and about 5% on average compared to the no-clarification baseline. Remarkably, this improvement is accompanied by a reduction in the average dialogue length by roughly two conversational turns. Our results clearly indicate that prompting agents to proactively clarify ambiguous, incomplete, or underspecified user requests consistently improves task success rates without extending dialogue length. This emphasizes the ben-

Method	Success Rate ( $\uparrow$ )
SimpleTOD (Hosseini-Asl et al., 2020)	22.00
UBAR (Yang et al., 2021)	26.80
GALAXY (Zhong et al., 2023)	28.80
MARS (Sun et al., 2023)	27.50
TOATOD (Bang et al., 2023)	26.90
FNCTOD (Li et al., 2024)	44.40
AutoTOD (Xu et al., 2024)	46.90
MAC	<b>58.40</b>

**Table 3: MAC’s Performance Compared to Existing TOD Systems.** Evaluation of various TOD methods using a standardized framework. Results for baseline models are sourced from AutoTOD to ensure fair and consistent comparison. Results taken from (Xu et al., 2024), following same evaluation protocol to ensure fairness in our comparison.

efit of proactive conversational strategies. These findings highlight the MAC framework’s superior performance, effectively balancing accuracy with conversational efficiency.

**Takeaway 1:** Asking clarification questions consistently **increase task success** and **decrease number of turns** to solve the task in multi-agent settings.

### 5.4 Ablation Studies

#### Comparison of MAC with Other TOD Systems

In Table 2, we demonstrated MAC’s performance compared to its variants, highlighting that combining clarification capabilities between supervisor and experts results in the most effective setup. To further contextualize MAC’s performance, it is crucial to benchmark against other leading task-oriented dialogue (TOD) systems. Following the evaluation framework of AutoTOD (Xu et al., 2024), we present this comparison in Table 3. Our results indicate that MAC surpasses previous state-of-the-art models, achieving an improvement of approximately 11.50% over the closest agent AutoTOD. This underscores both the robustness of

Language Model	Success Rate ( $\uparrow$ )
<b>MAC w/o Clarification</b>	
gpt-4o	53.72 $\pm$ 0.92
gpt-4o-mini	52.40 $\pm$ 2.08
Qwen3-235B-A22B	47.32 $\pm$ 1.72
<b>MAC with Clarification</b>	
gpt-4o	58.40 $\pm$ 2.10 (+4.68)
gpt-4o-mini	57.10 $\pm$ 1.42 (+4.70)
Qwen3-235B-A22B	54.50 $\pm$ 1.06 (+7.28)

**Table 4:** MAC success rates with different LLMs, with and without clarification. **Values in parentheses** show absolute improvement from clarification.

the multi-agent architecture in multi-domain scenarios such as MultiWOZ 2.4 and the critical role of proactive clarification when handling uncertainties <sup>1</sup>.

**Takeaway 2:** MAC demonstrates superior performance over previous TOD systems, attributed to its multi-agent architecture and effective user clarification capabilities.

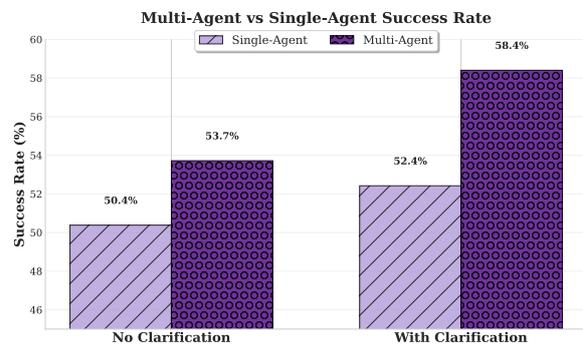
**How does the choice LLM effect MAC?** Since multi-agent setups are typically constructed using multiple LLMs with prompting, it is valuable to evaluate the performance of diverse LLMs within our MAC framework. To this end, we experimented with proprietary API-based and open-source models: GPT-4o-2024-11-20 (Hurst et al., 2024), gpt-4o-mini, Qwen3-235B-A22B (Yang et al., 2025). As shown in Table 4, enabling coordinated user clarification for both supervisor and expert agents in the MAC framework consistently improves task success rates, regardless of model type. For instance, gpt-4o and gpt-4o-mini achieve absolute improvements of +4.68 and +4.70 points, respectively, when equipped with clarification. Notably, the open-source Qwen3-235B-A22B model exhibits an even larger gain of +7.18 points, narrowing the gap with proprietary counterparts. The larger delta in accuracy for open-source LLMs suggests that well-designed supervision and agent coordination can unlock their potential, making them

<sup>1</sup>Some earlier TOD systems in Table 2 were developed prior to the integration of LLMs and follow fundamentally different pipelines (Acikgoz et al., 2025c), making direct comparisons not fully fair. Nevertheless, this comparison aims to contextualize MAC’s performance and also illustrate the overall progress of TOD systems over time.

competitive candidates with proprietary models for agentic systems in practice.

**Takeaway 3:** Enabling user clarification for both supervisor and expert agents in MAC consistently improves performance **regardless of model type**, even with open-source models.

**Takeaway 4:** Proactive interaction and effective agent coordination yield the highest accuracy gains for open-source LLMs, making them as strong alternatives to proprietary models in agentic systems.



**Figure 3:** Comparison of Single-Agent vs Multi-Agent Systems With and Without Clarification. The multi-agent approach benefits more from user clarification, achieving the highest performance.

## 5.5 MAC vs Single Agent Clarification

Figure 3 presents a comparative analysis of single-agent and multi-agent systems using GPT-4o-2024-11-20, examining their performance with and without user clarification. Our findings demonstrate that clarification enhances success rates in both setups; however, the improvement is notably more pronounced in the multi-agent configuration. Specifically, the Multi-Agent Clarification (MAC) system outperforms Single-Agent Clarification (SAC) by 6% (52.4%  $\rightarrow$  58.4%), highlighting the advantage of separating responsibilities between a Supervisor, responsible for general, high-level ambiguities, and domain-specific Experts who handle specialized clarifications. Moreover, our multi-agent setup consistently achieves higher success rates than the single-agent approach even in scenarios without clarification, emphasizing the inherent benefit of distributing workload among coordinated agents rather than overloading a single agent with multiple roles. This superior performance, particularly under clarification conditions, underscores the advantages of modularity

in resolving ambiguities effectively and efficiently scaling to complex, multi-domain interactions. In contrast, single-agent systems face diminishing returns and decreased interpretability as complexity grows due to the necessity of internalizing diverse expertise. Additionally, the modular structure of MAC facilitates incremental updates and streamlined integration of new domains, enhancing its robustness and practical applicability in real-world scenarios, as demonstrated by systems such as the model context protocol (MCP) (Hou et al., 2025).

**Takeaway 5:** Multi-agent setup outperforms single-agent both with and without clarification, which suggest **multi-agent setup offers improved scalability and efficiency** as each agent operates with a more focused and concise context.

Method	Success Rate ( $\uparrow$ )
MAC	58.40 $\pm$ 2.10
w/o Ambiguity & Vagueness Handling	52.20 $\pm$ 1.21 (-6.20)
w/o Slot/Parameter-Blocking Clarification	56.22 $\pm$ 1.36 (-2.18)

**Table 5:** Ablation study on the impact of supervisor (Ambiguity & Vagueness Handling) and expert (Slot/Parameter-Blocking Clarification) clarification taxonomies, with the drop in accuracy (**red**).

**Effect of Clarification Taxonomies on Task Success** To better understand the critical roles of different clarification strategies and understand the correspondence between the different ambiguity types defined in the taxonomy, we clustered the supervisor’s taxonomy into high-level Ambiguity and Vagueness Handling (encompassing Domain Ambiguity, Intent Ambiguity, Vague Goal Specification, and Contextual Disambiguation) and the expert’s taxonomy into Slot/Parameter Uncertainty (covering Parameter Underspecification and Value Ambiguity/Vagueness). As shown in Table 5 based on GPT-4o-2024-11-20 outputs, ablating Ambiguity and Vagueness Handling from the supervisor yields a substantial drop in task success rate ( $-6.20$ ), highlighting the importance of proactively resolving common-sense ambiguities before delegating to domain-specific experts. In contrast, removing Slot/Parameter-Blocking Clarification from experts causes a smaller, but still notable, accuracy decline ( $-2.18$ ), confirming that careful handling of underspecified or vague user slots is also essential, especially for API calls requiring precise parameters. It demonstrates that both forms of clarification

are necessary, but high-level disambiguation with supervisor is particularly crucial for robust multi-agent dialogue.

**Takeaway 6:** High-level **ambiguity and vagueness handling by the supervisor is essential** for robust performance, with its removal causing the largest drop in task success among all clarification skills.

Overall, these results support our preference for the multi-agent architecture, where MAC delivers higher accuracy, improved scalability, and enhanced maintainability, underscoring the critical role of modular design in developing robust and generalizable conversational agents.

## 6 Discussion

**Conclusions** We introduce MAC, the first multi-agent LLM framework specifically designed for interactive user clarification in conversational agents, and we also present the first comprehensive user clarification taxonomy for this domain, best of our knowledge. Our results demonstrate that effective user clarification is essential for maximizing task success and conversational efficiency, minimizing unnecessary user interactions. The proposed taxonomy enhances accuracy in both single-agent and multi-agent settings, with the multi-agent approach yielding superior results due to effective task sharing with domain experts. Notably, MAC is model-agnostic and significantly boosts performance across both open-source and proprietary LLMs, with particularly pronounced gains for open-source models; helping close the performance gap through optimal design and supervision. Our ablation studies highlight that high-level ambiguity and vagueness handling by the supervisor is especially critical for robust performance in real-world scenarios. Overall, MAC establishes a foundation for future research and deployment of multi-agent, user-centric conversational systems, offering clear benefits for practical and industrial applications.

**Limitations** Although MAC is a new effective and promising framework, it has several limitations. First, due to the complexity of multi-turn conversations, successful goal-oriented dialogue tasks and planning require large-scale capable models. As a result, our experiments primarily rely on large-parameter, API-based models. Teaching these capabilities to smaller LLMs and deploying them

efficiently remains an open challenge (Belcak et al., 2025). Second, we use LLM-as-a-Judge (LaaJ) based evaluations, which has known limitations in its core. In rare cases, the evaluator LLM may hallucinate and assign positive scores to incorrect dialogue trajectories; even though, we did not observe such failures, outlier cases may still exist. Nevertheless, LaaJ has been widely adopted in prior TOD research and has proven effective for evaluating complex, long-horizon conversational rollouts that are difficult to assess with deterministic metrics alone (Xu et al., 2024; Acikgoz et al., 2025b). Combining LaaJ with complementary deterministic, rule-based metrics could improve robustness, but designing such hybrid evaluations remains an open challenge. Finally, our error analysis reveals that in rare cases (approximately 1%), the agent and the user simulator can become stuck in a loop (Barres et al., 2025), where the agent repeatedly asks clarification questions and the simulator returns the same response. Although infrequent, this behavior highlights the need for more realistic user simulators and points to human-agent co-evolution as an interesting direction.

**Future Work** Handling user interactions in conversational agents is non-trivial, as it requires managing multiple tasks such as providing accurate responses or invoking the appropriate API from thousands of available tools (Su et al., 2025). In addition to these requirements, MAC further focus on challenge at scale through its carefully designed multi-agent setup, which is specifically tailored for user clarification. As a future direction, agents could learn optimal timing for seeking user clarification by monitoring environmental signals during interactions, leveraging recent reinforcement learning techniques (Lambert et al., 2024; Guo et al., 2025) to continuously self-update and become increasingly successful over time as a promising path towards self-improving agents (Zhang et al., 2025; Acikgoz et al., 2025d). Moreover, while our evaluation focuses on task success and the number of conversational turns as proxies for efficiency, quantifying overall user satisfaction remains an open question (Terry et al., 2023), where factors such as dialogue naturalness and other user-centric elements may play a pivotal role.

## References

Emre Can Acikgoz, Jeremiah Greer, Akul Datta, Ze Yang, William Zeng, Oussama Elachqar, Em-

manouil Koukoumidis, Dilek Hakkani-Tür, and Gokhan Tur. 2025a. [Can a single model master both multi-turn conversations and tool use?](#) *CoALM: A unified conversational agentic language model*. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12370–12390, Vienna, Austria. Association for Computational Linguistics.

Emre Can Acikgoz, Carl Guo, Suvodip Dey, Akul Datta, Takyoun Kim, Gokhan Tur, and Dilek Hakkani-Tur. 2025b. [TD-EVAL: Revisiting task-oriented dialogue evaluation by combining turn-level precision with dialogue-level comparisons](#). In *Proceedings of the 26th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 113–132, Avignon, France. Association for Computational Linguistics.

Emre Can Acikgoz, Dilek Hakkani-Tür, and Gokhan Tur. 2025c. [Conversational agents in the era of large language models \[perspectives\]](#). *IEEE Signal Processing Magazine*, 42(3):35–39.

Emre Can Acikgoz, Cheng Qian, Heng Ji, Dilek Hakkani-Tür, and Gokhan Tur. 2025d. [Self-improving llm agents at test-time](#). *arXiv preprint arXiv:2510.07841*.

Emre Can Acikgoz, Cheng Qian, Hongru Wang, Vardhan Dongre, Xiusi Chen, Heng Ji, Dilek Hakkani-Tür, and Gokhan Tur. 2025e. [A desideratum for conversational agents: Capabilities, challenges, and future directions](#). *arXiv preprint arXiv:2504.16939*.

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.

Chinmaya Andukuri, Jan-Philipp Fränken, Tobias Gerstenberg, and Noah D Goodman. 2024. [Star-gate: Teaching language models to ask clarifying questions](#). *arXiv preprint arXiv:2403.19154*.

Namo Bang, Jeehyun Lee, and Myoung-Wan Koo. 2023. [Task-optimized adapters for an end-to-end task-oriented dialogue system](#). In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7355–7369, Toronto, Canada. Association for Computational Linguistics.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. 2025.  [\$\tau^2\$ -bench: Evaluating conversational agents in a dual-control environment](#). *Preprint*, arXiv:2506.07982.

Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Saurav Muralidharan, Yingyan Celine Lin, and Pavlo Molchanov. 2025. [Small language models are the future of agentic ai](#). *arXiv preprint arXiv:2506.02153*.

Willy Chung, Samuel Cahyawijaya, Bryan Wilie, Holy Lovenia, and Pascale Fung. 2023. [InstructTODS:](#)

- Large language models for end-to-end task-oriented dialogue systems. In *Proceedings of the Second Workshop on Natural Language Interfaces*, pages 1–21, Bali, Indonesia. Association for Computational Linguistics.
- Yang Deng, Wenqiang Lei, Wai Lam, and Tat-Seng Chua. 2023. A survey on proactive dialogue systems: problems, methods, and prospects. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pages 6583–6591.
- Vardhan Dongre, Xiaocheng Yang, Emre Can Acikgoz, Suvodip Dey, Gokhan Tur, and Dilek Hakkani-Tür. 2024. Respect: Harmonizing reasoning, speaking, and acting towards building large language model-based conversational ai agents. *arXiv preprint arXiv:2411.00927*.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.
- Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xi-angliang Zhang. 2024. Large language model based multi-agents: a survey of progress and challenges. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI '24*.
- Aman Gupta, Anirudh Ravichandran, Narayanan Sadagopan, and Anurag Beniwal. 2024. DARD: A multi-agent approach for task-oriented dialog systems. In *NeurIPS 2024 Workshop on Open-World Agents*.
- Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. In *Advances in Neural Information Processing Systems*, volume 33, pages 20179–20191. Curran Associates, Inc.
- Xinyi Hou, Yanjie Zhao, Shenao Wang, and Haoyu Wang. 2025. Model context protocol (mcp): Landscape, security threats, and future research directions. *arXiv preprint arXiv:2503.23278*.
- Yushi Hu, Chia-Hsuan Lee, Tianbao Xie, Tao Yu, Noah A. Smith, and Mari Ostendorf. 2022. In-context learning for few-shot dialogue state tracking. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2627–2643, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Vojtěch Hudeček and Ondrej Dusek. 2023. Are large language models all you need for task-oriented dialogue? In *Proceedings of the 24th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 216–228, Prague, Czechia. Association for Computational Linguistics.
- Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Os-trow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.
- Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2022. Clam: Selective clarification for ambiguous questions with generative language models. *arXiv preprint arXiv:2212.07769*.
- Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. 2024. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*.
- Belinda Z Li, Alex Tamkin, Noah Goodman, and Jacob Andreas. 2023. Eliciting human preferences with language models. *arXiv preprint arXiv:2310.11589*.
- Zekun Li, Zhiyu Chen, Mike Ross, Patrick Huber, Seungwhan Moon, Zhaojiang Lin, Xin Dong, Adithya Sagar, Xifeng Yan, and Paul Crook. 2024. Large language models as zero-shot dialogue state tracker through function calling. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8688–8704, Bangkok, Thailand. Association for Computational Linguistics.
- Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. 2025. RouteLLM: Learning to route LLMs from preference data. In *The Thirteenth International Conference on Learning Representations*.
- Hongjin Su, Shizhe Diao, Ximing Lu, Mingjie Liu, Ji-acheng Xu, Xin Dong, Yonggan Fu, Peter Belcak, Hanrong Ye, Hongxu Yin, et al. 2025. Toolorchestra: Elevating intelligence via efficient model and tool orchestration. *arXiv preprint arXiv:2511.21689*.
- Haipeng Sun, Junwei Bao, Youzheng Wu, and Xiaodong He. 2023. Mars: Modeling context & state representations with contrastive learning for end-to-end task-oriented dialog. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11139–11160, Toronto, Canada. Association for Computational Linguistics.
- Lijun Sun, Yijun Yang, Qiqi Duan, Yuhui Shi, Chao Lyu, Yu-Cheng Chang, Chin-Teng Lin, and Yang Shen. 2025. Multi-agent coordination across diverse applications: A survey. *arXiv preprint arXiv:2502.14743*.
- Michael Terry, Chinmay Kulkarni, Martin Wattenberg, Lucas Dixon, and Meredith Ringel Morris. 2023. Interactive ai alignment: Specification, process, and evaluation alignment. *arXiv preprint arXiv:2311.00710*.
- Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O’Sullivan, and Hoang D

- Nguyen. 2025. Multi-agent collaboration mechanisms: A survey of llms. *arXiv preprint arXiv:2501.06322*.
- Heng-Da Xu, Xian-Ling Mao, Puhai Yang, Fanshu Sun, and Heyan Huang. 2024. [Rethinking task-oriented dialogue systems: From complex modularity to zero-shot autonomous agent](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2748–2763, Bangkok, Thailand. Association for Computational Linguistics.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.
- Yunyi Yang, Yunhao Li, and Xiaojun Quan. 2021. Ubar: Towards fully end-to-end task-oriented dialog system with gpt-2. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 14230–14238.
- Fanghua Ye, Jarana Manotumruksa, and Emine Yilmaz. 2022. [MultiWOZ 2.4: A multi-domain task-oriented dialogue dataset with essential annotation corrections to improve state tracking evaluation](#). In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 351–360, Edinburgh, UK. Association for Computational Linguistics.
- Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. 2025. Darwin godel machine: Open-ended evolution of self-improving agents. *arXiv preprint arXiv:2505.22954*.
- Michael JQ Zhang and Eunsol Choi. 2023. Clarify when necessary: Resolving ambiguity through interaction with llms. *arXiv preprint arXiv:2311.09469*.
- Michael JQ Zhang and Eunsol Choi. 2025. [Clarify when necessary: Resolving ambiguity through interaction with LMs](#). In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5526–5543, Albuquerque, New Mexico. Association for Computational Linguistics.
- Michael JQ Zhang, W Bradley Knox, and Eunsol Choi. 2024. Modeling future conversation turns to teach llms to ask clarifying questions. *arXiv preprint arXiv:2410.13788*.
- Lucen Zhong, Hengtong Lu, Caixia Yuan, Xiaojie Wang, Jiashen Sun, Ke Zeng, and Guanglu Wan. 2023. A task-oriented dialog model with task-progressive and policy-aware pre-training. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 3–15. Springer.

## A Further Details on Evaluation and MultiWOZ 2.4

We evaluate the performance of our MAC framework using dialogue-level metrics that capture both the effectiveness and efficiency of task completion. Our primary metric is Success Rate, which measures whether the agent fully satisfies all user-specified constraints and successfully completes the task. For each dialogue, we use an LLM-based judge using GPT-4o-2024-11-20 (Hurst et al., 2024) to assess if the agent’s final response fulfills every requirement defined by the user’s goal, including both requested attributes (such as hotel name or train arrival time) and booking constraints (such as the number of people or destination) following Xu et al. (2024). Formally, a dialogue is considered successful if all constraints in the user’s goal  $G$  are met by the end of the interaction:

$$\text{Success} = \mathbb{I}(\text{all constraints in } G \text{ are satisfied}), \quad (1)$$

where  $\mathbb{I}(\cdot)$  denotes the indicator function. This score is computed for every dialogue and averaged across the evaluation set. To account for the stochastic nature of both model inference and LLM-based judging, we conduct five independent runs for each experimental configuration.

We report two aggregate Success Rate metrics: **Success Max@5**, the highest single-run success rate out of five runs, reflecting the best-case performance; and **Success Avg@5**, the mean and standard deviation of success rates over the five runs, providing a robust measure of typical performance and variance. In addition, we report the **Average Number of Turns** per conversation as an efficiency metric. This measures the average length of the dialogue required to complete the task, with lower values indicating more concise and effective interactions. This metric is particularly important for assessing the practical impact of clarification strategies on user burden and overall system efficiency.

## B Experimental Details

During our experiments, we use the OpenAI API<sup>2</sup> to evaluate the GPT models GPT-4o-2024-11-20 (Hurst et al., 2024) and gpt-4o-mini. For open-source model evaluations, we use the TogetherAI API<sup>3</sup> with Qwen3-235B-A22B (Yang et al., 2025). To ensure

reproducibility, we use default generation settings for all models without tuning any inference hyperparameters. In terms of runtime, a single model inference takes approximately 15 minutes, while evaluation requires around 4–5 minutes. Since LLM-based judge metrics are non-deterministic, the results may vary slightly across runs. To account for this variability, we perform all evaluations five times and report scores with standard deviations.

<sup>2</sup><https://openai.com/api/>

<sup>3</sup><https://api.together.xyz/>

Domain	API Name	API Arguments	Test Samples per Domain
Restaurant	query_restaurant	area, pricerange, food, name	437
	book_restaurant	name, people, day, time, pricerange, stars, type	
Hotel	query_hotel	area, internet, name, parking	394
	book_hotel	name, people, day, stay	
Attraction	query_attraction	area, name, type	395
Train	query_train	arriveBy, day, departure, destination, leaveAt, trainID	494
	buy_train_ticket	arriveBy, day, departure, destination, leaveAt, trainID, people	
Taxi	book_taxi	arriveBy, departure, destination, leaveAt	195

**Table 6:** Available actions per domain in MultiWOZ 2.4 dataset.

### ClarifyRLData Generation Prompt

Given the following User Goal and Ground Truth Conversation, update the conversation to introduce ambiguity or underspecification in the user’s turns, such that the agent must ask for clarification **at least once and at most three times**. For every agent clarification, enclose the clarification in ‘<clarify>’ and ‘</clarify>’ tokens. After each agent clarification, update the following user turn(s) to resolve the ambiguity. Do not change the overall goal or successful task completion. Only modify the conversation for clarification needs.

#### Task:

- Carefully read the User Goal and the Ground Truth Conversation.
- Rewrite the conversation so that some user turns are ambiguous or missing key information, requiring the agent to clarify at least once and at most three times (with the ‘<clarify>’ tokens).
- Keep the rest of the conversation as natural as possible and ensure the final output still accomplishes the user goal.

#### Output Format:

- Output the **updated conversation only** as a **valid JSON array** in the format:

```
““json
{
  "from": "user", "value": "...",
  "from": "agent", "value": "<clarify>...</clarify>",
  "from": "user", "value": "...",
  "from": "agent", "value": "...",
  ...
}
““
```

- **No extra text, no comments, no explanations, no markdown—just the JSON array.**
- The output **must** be valid JSON.
- If the format is not exactly correct, data loading will fail.

#### User Goal

<user\_goal>

#### Ground Truth Conversation

<conversation>

#### New Conversation with User Clarification

{ Your JSON Data Here }

**Figure 4: ClarifyRL Data Generation Prompt.** Prompt used to synthesize ClarifyRL data with an LLM, conditioned on a given user goal and dialogue from the MultiWOZ 2.4 training split.

## Supervisor Agent Prompt

### ### Task

You are an expert routing agent in a multi-domain conversational AI system for the MultiWOZ dataset. Your specific task is to analyze a user's query and determine which of the following five domain experts is best suited to handle it:

1. restaurant (for queries related to finding, booking, or getting information about restaurants)
2. hotel (for queries related to finding, booking, or getting information about hotels or other accommodations)
3. attraction (for queries related to finding or getting information about tourist attractions, landmarks, or points of interest)
4. train (for queries related to finding, booking, or getting information about train travel)
5. taxi (for queries related to booking or getting information about taxi services)

Read the user's query provided below (### User Query). Your goal is to identify the single, most dominant domain relevant to the query.

### ### Output Instructions

You MUST output ONLY the exact lowercase label corresponding to the selected domain, enclosed in <domain> and </domain>.

For example, if the query is about a hotel, your output must be <domain>hotel</domain>. Do NOT include any other words, phrases, explanations, or punctuation outside of these tags. Your entire response should be just one of these five labels, wrapped in the domain tags as shown below:

```
<domain>restaurant</domain>
<domain>hotel</domain>
<domain>attraction</domain>
<domain>train</domain>
<domain>taxi</domain>
```

If the query seems to touch on multiple domains, select the one that appears to be the primary focus or the one that needs to be addressed first.

If no domain is clearly identifiable from the list, you must still choose the closest possible one or a default agreed upon (though for this specific instruction, you must pick one of the five and wrap it in the tags).

### ### User Query

```
{{user_query}}
```

Selected Domain Label:

**Figure 5: LLM Prompt** used for the Supervisor agent to decide domain routing.

## Hotel Domain Expert Prompt

### ### Role Description

You are an advanced AI assistant specializing in conversational dialogues focused on the hotel domain. You can act both as a system (providing hotel information and booking services) and a user (interacting with the hotel database) to assist users in completing hotel-related tasks.

### ### Task Information

- Each time, you must determine whether to call an API by reasoning through "Thought:".
- If you decide that an API call is necessary, include "Thought:" for reasoning, followed by "API Name:", "API Input:", "API Result:".
- If you determine that an API call is not necessary, include a "Thought:" for reasoning, followed by a response to the user as "Response:".
- If the user asks for some attributes of a hotel (e.g., address, phone number, price range, parking, internet), then an API call is necessary.
- You are not allowed to use APIs not mentioned below. If you decide that the mentioned APIs are not sufficient for the user's request, you should inform the user that you can only assist with hotel queries and bookings.
- If you decide that more than one API call is needed (e.g., query first, then book), you should call one API first and wait for the API result. After obtaining that result, you may think and call the next API or think and make a response.
- The user can sometimes not care about the value of an API input slot and may mention it explicitly in the conversation (e.g., "I don't care about the price range"). In such cases, predict "dontcare" as a slot value for that particular slot.
- If you decide that there is an API input slot that the user has never mentioned and is required for the API, please put "any" as the slot value as a placeholder.
- You can put only one value in each API input slot per query.

### ATTENTION:

- Predict "dontcare" as a slot value ONLY if the user has explicitly mentioned it in the conversation.

### ### Output Format

- If an API Call is Needed:
  - Thought: [Your reasoning for why an API call is needed]
  - API Name: [Available APIs: query\_hotels, book\_hotel]
  - API Input: [The input parameters for the API as a JSON]
  - API Result:
- If an API Call is Not Needed:
  - Thought: [Your reasoning for why an API call is not needed and you are responding directly]
  - Response: [Your response to the user]

### ### API Details:

{{api\_descriptions}}

### ### Example with explanation

{{example\_conversation}}

Figure 6: LLM Prompt used for the Domain Expert agent to decide response.

## Supervisor Agent Prompt with User Clarification

### ### Task

You are a high-level supervisor and routing agent in a multi-domain conversational AI system. Your primary goal is to analyze a user's query and take one of two actions:

1. **Clarify:** If the user's intent or desired domain is ambiguous, ask a single, precise clarification question.
2. **Route:** If the user's intent and domain are clear, route the query to the appropriate expert agent.

### ### Core Directive

Your responsibility is to handle ONLY high-level, common-sense, and domain-agnostic ambiguities. **DO NOT** ask for domain-specific details (e.g., cuisine, price range, number of people, time of booking). Your task is to figure out **WHAT** the user wants in general, not the specifics of how to do it.

### ### Clarification Taxonomy (When to Ask)

Before routing, you must determine if clarification is needed. Use this taxonomy to guide your decision:

- **Domain Ambiguity:** The query could fit multiple domains (e.g., "Find me a good place." - good place to eat or stay?).
  - **Intent Ambiguity:** The domain is clear, but the user's goal is not (e.g., "Tell me about trains" - find a schedule or book a ticket?).
  - **Vague Goal Specification:** The query is too broad to be actionable (e.g., "Help me with my trip" - what kind of help; booking or search?).
  - **Contextual Disambiguation:** The query uses vague references like "it" or "that place" which are unclear from the context.
  - **General Conflict:** Broad contradictions in user's input not domain-specific (e.g., "I changed my mind about the date").
  - **General Noise/Correction:** Common errors or typos needing clarification (e.g., "I meant tomorrow not today").
  - **Unfamiliar Domain Request:** Request does not match known domains clearly (e.g., "Can you help me fix my phone?" - no such a domain or expert).
- If a clarification is needed, **always output a clarifying question in the format:**
- Thought: The user request is unclear due to [reason].
  - Response: <clarify>[Your response question to the user for clarification]</clarify>

### ### Output Instructions

1. For user clarification, you should provide your reasoning as "Thought: [your reasoning]" and your user clarification question response as "Response: <clarify>[your high-level clarification question]</clarify>".
2. For domain selection, you should provide your response between <domain> and </domain> tags.

Example:

### ### Output Format

You MUST output either a '<clarify>' tag OR a '<domain>' tag in a single turn.

#### 1. If User Clarification is Needed:

Thought: [Your reasoning for why user clarification is needed, you are responding directly]

Response: <clarify>[Your response question to the user]</clarify>

#### 2. If The Query is Clear and Routable:

Your entire output must be the single, lowercase domain label wrapped in the '<domain>' tag.

You MUST output ONLY the exact lowercase label corresponding to the selected domain, enclosed in <domain> and </domain>.

For example, if the query is about a hotel, your output must be <domain>hotel</domain>. Your entire response should be just one of these five labels, wrapped in the domain tags as shown below:

- <domain>restaurant</domain>
- <domain>hotel</domain>
- <domain>attraction</domain>
- <domain>train</domain>
- <domain>taxi</domain>

If the query seems to touch on multiple domains, select the one that appears to be the primary focus or the one that needs to be addressed first. If no domain is clearly identifiable from the list, you must still choose the closest possible one or a default agreed upon (though for this specific instruction, you must pick one of the five and wrap it in the tags).

### ### Conversation History

conversation\_history

### ### User Query

user\_query

### ### Output:

Figure 7: LLM Prompt used for the Supervisor agent to decide domain routing and user clarification.

## Hotel Expert Prompt with User Clarification

### ### Role Description

You are an advanced AI assistant specializing in conversational dialogues focused on the hotel domain.

You can act both as a system (providing hotel information and booking services) and a user (interacting with the hotel database) to assist users in completing hotel-related tasks.

### ### Task Information

- Each time, you must determine whether to call an API by reasoning through "Thought:".
- If you decide that an API call is necessary, include "Thought:" for reasoning, followed by "API Name:", "API Input:", "API Result:". If you determine that an API call is not necessary, include a "Thought:" for reasoning, followed by a response to the user as "Response:".
- If the user asks for some attributes of a hotel (e.g., address, phone number, price range, parking, internet), then an API call is necessary.
- You are not allowed to use APIs not mentioned below. If you decide that the mentioned APIs are not sufficient for the user's request, you should inform the user that you can only assist with hotel queries and bookings.
- If you decide that more than one API call is needed (e.g., query first, then book), you should call one API first and wait for the API result. After obtaining that result, you may think and call the next API or think and make a response. The user can sometimes not care about the value of an API input slot and may mention it explicitly in the conversation (e.g., "I don't care about the price range"). In such cases, predict "dontcare" as a slot value for that particular slot.
- If you decide that there is an API input slot that the user has never mentioned and is required for the API, please put "any" as the slot value as a placeholder. You can put only one value in each API input slot per query.

### ### Clarification Taxonomy (When to Ask)

Before calling an API, determine if you have all the necessary information. If not, ask a clarifying question using this taxonomy:

- **Parameter Underspecification:** Key details for a search or booking are missing (e.g., location, cuisine, number of people, time).
- **Value Ambiguity/Vagueness:** A user's term is subjective and needs clarification (e.g., "a nice place," "somewhere soon").
- **Constraint Conflict:** The user provides contradictory information (e.g., "a cheap but expensive restaurant").
- **Entity Disambiguation/Not Found:** A specific restaurant name is ambiguous or cannot be found.
- **Confirmation of Inferred Information:** You have inferred a detail from context and need to confirm it before proceeding.
- It is also important not to burden the user with repetitive or similar clarification questions in your overall conversation; please be mindful of this during your conversation.
- If a clarification is needed, **always output a clarifying question in the format:**
  - Thought: The user request is unclear due to [reason].
  - Response: <clarify>[Your response question to the user for clarification]</clarify>
- If you decided to ask the user for further clarification about the user query, you should output your user clarification question as: <clarify>...</clarify>. Your output should be like this:

Thought: [Your reasoning for asking user clarification questions]

Response: <clarify>YOUR RESPONSE FOR USER CLARIFICATION HERE</clarify>

### ### Output Format

- If an API Call is Needed:

Thought: [Your reasoning for why an API call is needed]

API Name: [Available APIs: query\_hotels, book\_hotel]

API Input: [The input parameters for the API as a JSON]

API Result:

- If an API Call is Not Needed:

Thought: [Your reasoning for why an API call is not needed and you are responding directly]

Response: [Your response to the user]

### ### API Details:

{{api\_descriptions}}

### ### Example with explanation

{{example\_conversation}}

Figure 8: LLM Prompt used for the Supervisor agent to decide domain routing and user clarification.