Towards Preference Following in Tool Calling Language Agents

Anonymous ACL submission

Abstract

Large language model (LLM)-based agents have demonstrated remarkable capabilities in tool use, but their ability to follow user preferences when calling tools remains underexplored. To address this gap, we introduce APOLLO, a benchmark designed to evaluate agents' ability to identify personalized user preferences from interaction histories and to adhere to these preferences when calling tools to solve user queries. In APOLLO, user preferences expressed in the interaction history take two forms: explicit through direct statements and implicit through option selections. In addition, the benchmark includes two types of queries, reactive and proactive, which pose challenges for LLMs to ground user queries in the corresponding preferences. Using APOLLO, we evaluate and analyze both language models and reasoning models, and investigate the impact of different agent frameworks, such as Reflexion, on model performance. Experimental results show that current models still struggle to follow user preferences when calling tools. For instance, GPT-4o achieves only 51.34% accuracy on the benchmark. Furthermore, we develop a reinforcement learning-based approach to improve LLMs, achieving substantial performance gains on APOLLO. Our dataset and code are publicly available at `https://anonymous.4open.science/r/APOLLO_anony`.

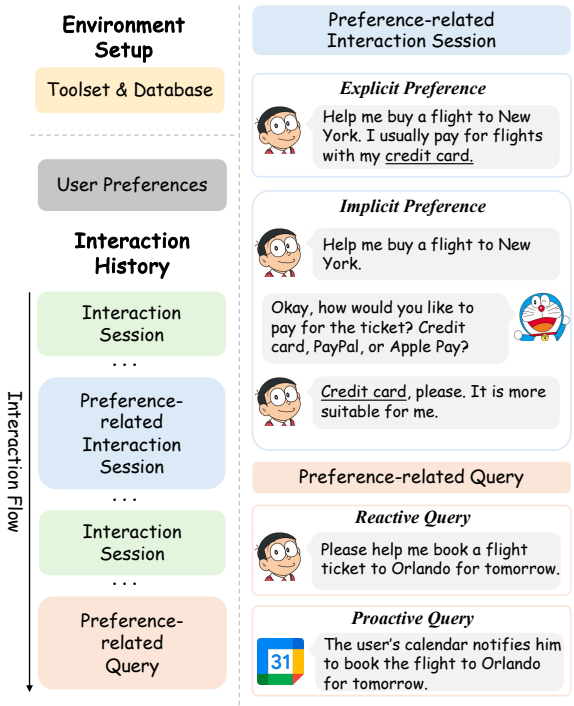

Figure 1: Overview of the APOLLO benchmark. Each benchmark sample consists of a multi-session user–agent interaction history, where the user reveals a preference in one of the sessions, along with a query related to that preference. User preferences are expressed in two forms-explicitly or implicitly-while queries fall into two categories: reactive and proactive. APOLLO evaluates the ability of LLMs to follow the corresponding user preferences when addressing the queries.

1 Introduction

Tool calling is one of the core capabilities of large language model (LLM)-based agents (Schick et al., 2023; Qin et al., 2023; Wang et al., 2024; Xi et al., 2025). By incorporating tool descriptions, parameter specifications, and related information into their inputs, LLMs can autonomously invoke appropriate tools to achieve goals such as data processing (Shen et al., 2023; Wu et al., 2024b), information retrieval (Gao et al., 2023b; Qiao et al., 2025), and coding (Gao et al., 2023a; Feng et al., 2025). Existing work on tool calling has primarily focused on the task side, aiming to improve the accuracy (Liu et al., 2024a; Jin et al., 2025), safety (Ye et al., 2024; Chen et al., 2024b), and efficiency (Huang et al., 2023) of LLM-based agents in tool use for general users. As agents continue to evolve and find broader applications, researchers have increasingly shifted attention to the **user side**, where the central challenge lies in aligning agents with personalized user needs (Salemi et al., 2023;

Zhao et al., 2025). In this new stage, following user preferences emerges as an unavoidable challenge.

To effectively follow user preferences, LLMs need to autonomously infer a user's personalized preferences from the interaction history and adhere to these preferences when calling tools to solve user queries. For example, as illustrated in Figure 1, a user mentions in the interaction history: *I usually pay for flights with my credit card*. Ideally, when the agent later uses a payment tool to help the user purchase a flight ticket, it should prioritize the credit card option. While ignoring this preference may still result in successful task completion (e.g., paying via PayPal), it substantially degrades the user experience, ultimately reducing satisfaction and willingness to adopt such agents (Salemi et al., 2023; Jiang et al., 2025). Despite the critical importance of preference following for LLM-based agents, there remains a lack of a systematic benchmark to evaluate their ability in this regard.

Evaluating the ability of agents to follow user preferences presents unique challenges. Real-world user-agent interactions often involve rich, multi-turn sessions, where preferences must be recalled over long contexts (Jiang et al., 2025; Hao et al., 2025). Moreover, users express their preferences in diverse forms (Zhao et al., 2025), and their queries vary in type (Lu et al., 2024; Yang et al., 2025b), both of which make it difficult for LLMs to correctly interpret the underlying requirements and faithfully ground these requirements in the relevant preferences. To address these challenges, we introduce **APOLLO** (Agent Preference fOLLOwing), a benchmark consisting of $2,080$ carefully curated samples. Each sample contains a multi-session interaction history, with context lengths ranging from $2k$ to $12k$ tokens. In APOLLO, user preferences are expressed in two ways: explicitly stated by the user or implicitly revealed through option selections. Furthermore, user queries are categorized into reactive and proactive types. When calling tools, LLMs are required to either map user preferences to the corresponding tool parameters or extract relevant information from tool outputs in light of the preferences to guide subsequent actions.

Using APOLLO, we evaluate 10 advanced models, encompassing both LLMs and large reasoning models (LRMs) (Jaech et al., 2024; Yang et al., 2025a). Our experiments reveal that current models struggle to consistently follow user preferences. For instance, GPT-OSS-120B and GPT-4o achieve only 32.89% and 51.34% accuracy, respectively, when averaged across different interaction history lengths. To investigate the reasons behind these failures, we conduct an error analysis and find that LLMs face challenges at multiple stages, including recognizing user preferences expressed in the interaction history and accurately adhering to them without hallucination. Furthermore, we examine whether prevalent agent frameworks, such as Reflexion (Shinn et al., 2023) and retrieval-augmented generation (RAG), can improve preference following, but the gains are limited (e.g., Qwen3-8B achieves an average accuracy improvement of 4.39% under the RAG framework compared to standard Function Calling.). Finally, to enhance the model's ability to follow user preferences, we develop an RL-based training approach, which substantially boosts performance, achieving 72.48% accuracy on the test set with an $8k$ context length.

2 The APOLLO Benchmark

2.1 Task Formulation

The overall structure of the APOLLO benchmark is illustrated in Figure 1. Formally, let H denote the interaction history between a user and an agent, consisting of n sessions, i.e., $H = \{s_1, s_2, \ldots, s_n\}$, where each session contains multi-turn interactions that may span different topics. Within this history, the user expresses a preference p in a particular session $s_p \in H$. Given a toolset T, the interaction history H, and a user query q that is relevant to the preference p, the goal of the APOLLO benchmark is to evaluate whether an LLM-based agent can identify and follow the user's preference p when invoking tools to resolve the query q.

It is important to note that the information about preference p appears only in session s_p, while the other sessions in H are unrelated to the query q and may distract the LLM from identifying and adhering to the correct preference. This design simulates realistic user-agent interactions, where different sessions often span multiple topics, posing a significant challenge for LLMs to accurately follow user preferences.

2.2 Benchmark Characteristics

Expression of Preferences. Following PREFEVAL (Zhao et al., 2025), our benchmark considers two distinct ways for users to express their preferences: (1) Explicit Expression. The user directly

states their preference during the interaction. (2) Implicit Selection. The user first issues a query related to their preference (e.g., *I'm traveling to New York this weekend. Help me buy a flight ticket to New York.*). In response, the agent presents several options corresponding to different possible preferences (e.g., *How would you like to pay for the ticket? Credit card, PayPal, or Apple Pay?*). One of these options aligns with the user's actual preference, and by selecting it, the user implicitly conveys their preference. Additional examples are provided in Appendix A.3.

Types of Queries. User queries in APOLLO fall into two types: reactive queries and proactive queries. A reactive query explicitly expresses the user's request in the first person, making the task straightforward for the LLM to understand and execute. In contrast, a proactive query describes the state of the environment rather than directly stating the task. The LLM must proactively infer the user's latent task from this contextual information and then call the appropriate tools to fulfill the user's needs. More query examples are shown in Appendix A.3.

Preference-Guided Tool Parameter Selection. In APOLLO, user preferences influence tool parameter selection in two distinct ways: (1) Direct Mapping of Preferences to Tool Parameters. In this setup, a specific parameter of the tool directly reflects the user's preference, and the LLM simply fills in the parameter with the corresponding preference value. For example, in the `search_movie` tool, the parameter `genre` specifies the type of movie to search for. If the user prefers romance movies, the LLM should set `genre = "romance"`. (2) Indirect Constraint of Tool Parameters. In this setup, the LLM must first call one tool to retrieve a list of potential parameter candidates for a subsequent tool. The user's preference then determines which candidate should be selected. For example, the `order_beverage` tool requires a parameter `beverage_id`. To determine this value, the LLM first calls the `search_beverage_id`, which returns a list of beverage IDs paired with their categories, e.g., `[{"BE813": "coffee"}, {"BE588": "juice"}]`. If the user prefers coffee, the LLM should select `"BE813"` as the value for `beverage_id` when calling `order_beverage`.

Challenges. The challenges posed by APOLLO lie in four key aspects: (1) Preference Identification. LLMs must accurately attend to and infer the user's preference from the interaction history, where the preference may be expressed either explicitly or implicitly. (2) Preference Adherence. LLMs are required to faithfully follow the inferred preference by filling in the appropriate tool parameters when calling tools to resolve the user's query. (3) Query Understanding. LLMs must correctly interpret user queries, which can be either reactive or proactive in form, and establish the connection between the query and the corresponding preference. (4) Long Context Horizon. Since the interaction history includes other unrelated sessions besides the one containing the relevant preference, these irrelevant sessions may distract LLMs. To successfully resolve the query, LLMs must possess a strong ability to maintain a long context horizon, enabling them to resist such distractions and accurately adhere to the user's preference.

2.3 Benchmark Construction

We provide a flowchart of the benchmark construction process in Figure 2.

Environment Setup. We manually constructed 15 tools spanning 4 domains. These domains cover common scenarios in which users frequently use tools (see Appendix A.1). To eliminate the influence of external factors, such as network latency or tool instability, we associate each tool with a local database. The tools retrieve their outputs directly from the corresponding database, ensuring that tool outputs remain stable and consistent throughout the benchmark.

Preference-Related Session and Query Construction. Our benchmark includes 20 distinct users. We first define each user's profile and preferences related to each domain, which jointly guide the generation of the subsequent user-agent interaction history. The profile captures general personal attributes of the user, while the preferences represent specific personal information related to the domains. To ensure diversity among user profiles, each profile is assigned a persona from PersonaHub (Ge et al., 2024) and augmented with demographic information such as gender and age. For preferences, we first manually construct several example preferences. These manually curated examples, along with the user profile, are then used by GPT-4.1 to generate domain-specific preferences for all users. Finally, we conduct human verification to ensure that the LLM-generated preferences

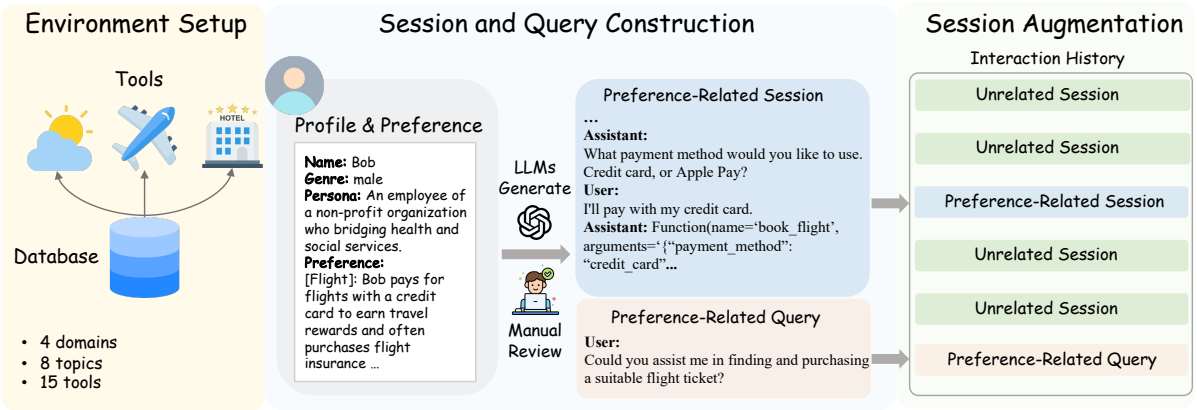

Figure 2: Flowchart of the APOLLO construction process.

can be faithfully reflected through tool usage.

Based on each preference, we construct the corresponding interaction sessions. Specifically, we begin by manually annotating an example interaction session grounded in one user's preference for each domain. This example, together with a user preference and a set of augmentation rules, is provided as input to GPT-4.1. The model then generates additional preference-related interaction sessions for the user. Finally, we perform a manual review to verify both the correctness and the preference alignment of tool calls in each generated session. The detailed prompt is presented in Appendix A.4.

In parallel, for each preference, we prompt GPT-4.1 to generate queries using manually created query examples as references. Each generated query undergoes human review to ensure that no information about the user's preference is leaked in the query. We then pair each verified query with its corresponding interaction session, resulting in a total of $2,080$ session-query pairs.

Session Augmentation. In addition to the sessions related to the 4 domain preferences described above, each user's interaction history also includes sessions unrelated to these preferences (see Section 2.1). To generate these unrelated sessions, we construct 7 additional domains with 31 tools, ensuring no overlap with the preference domains. The user profile, combined with these newly defined tools, is then provided as input to GPT-4.1, which generates the unrelated interaction sessions. Incorporating the user profile helps maintain consistency both within the unrelated sessions and between unrelated and preference-related sessions.

After generating the unrelated sessions, we randomly insert each preference-related session among these unrelated sessions to construct the final interaction history for a user. Each preference-related session is then paired with its corresponding query to form interaction history-query pairs. For each constructed interaction history, we select a tool set consisting of 8 tools, which includes the tools required to solve the query and randomly sampled tools that appear elsewhere in the history. Depending on the number of unrelated sessions, the total length of the context ranges from $2k$ to $12k$ tokens.

Labeling and Reward Definition. For each interaction history-query pair, we either manually annotate or automatically extract the tool parameters from the history to obtain the gold tool calling sequence, denoted as T_g. For each model-generated response trajectory, we extract the corresponding tool calling sequence T_t and compare it with T_g to compute the reward.

Our benchmark provides two reward metrics: **accuracy** and **F1 score**. For accuracy, the reward is set to 1 if T_t exactly matches T_g; otherwise, it is set to 0. For the F1 score, we first compute precision and recall. Let $N_{t\to g}$ and $N_{g\to t}$ denote the number of tools (including parameters) in T_t that also appear in T_g, and the number of tools in T_g that also appear in T_t, respectively. Let N_t and N_g represent the lengths of T_t and T_g. The precision and recall are then defined as: Precision $= N_{t\to g}/N_t$ and Recall $= N_{g\to t}/N_g$, respectively. Finally, the F1 score is calculated as the harmonic mean of precision and recall.

3 Experiments

3.1 Models and Agent Frameworks

We evaluate both LLMs and LRMs on our benchmark. For LLMs, we include Llama-3.1-8B-

Model	$2k$		$4k$		$8k$		$12k$	
	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
Llama-3.1-8B-Instruct	0.2933	0.5381	0.2202	0.4511	0.1769	0.4212	0.1606	0.4177
Qwen3-8B	0.4279	0.6445	0.3688	0.5808	0.3034	0.5222	0.2582	0.4757
Qwen3-8B (think)	0.4577	0.7118	0.4654	0.6850	0.4524	0.6731	0.3736	0.5973
Qwen3-30B-A3B	0.4163	0.6086	0.3024	0.4822	0.2913	0.4520	0.2500	0.4082
Qwen3-30B-A3B (think)	0.3543	0.6154	0.3130	0.5429	0.3077	0.5530	0.2596	0.5001
Qwen2.5-72B-Instruct	0.5115	0.7080	0.4202	0.6441	0.2889	0.4637	0.1861	0.3200
GPT-OSS-120B (think)	0.4183	0.6065	0.3514	0.5391	0.2702	0.4586	0.2755	0.4454
Gemini-2.5-Flash (think)	0.3928	0.6089	0.3096	0.5562	0.2707	0.5285	0.2519	0.5008
GPT-4o-mini	0.4764	0.6347	0.3971	0.5896	0.3678	0.5649	0.3663	0.5626
GPT-4o	**0.5726**	**0.7487**	**0.5034**	**0.7123**	**0.4947**	**0.7093**	**0.4827**	**0.6996**

Table 1: Performance of different models on APOLLO under the Function Calling framework. "(think)" indicates that the model explicitly outputs reasoning content when solving user queries (LRM). We report accuracy (Acc.) and F1 score (F1) in the table. The best results are highlighted in **bold**, and the second-best results are underlined.

Instruct (Dubey et al., 2024), Qwen3-8B, Qwen3-30B-A3B (Yang et al., 2025a), Qwen2.5-72B-Instruct (Team, 2024), GPT-4o-mini, and GPT-4o (Hurst et al., 2024). For LRMs, we include Qwen3-8B, Qwen3-30B-A3B, GPT-OSS-120B (Agarwal et al., 2025), and Gemini-2.5-Flash (Comanici et al., 2025), where Qwen3-8B and Qwen3-30B-A3B are switched to think mode by adding `<think>` in the prompt. For all models, we set the temperature to 0 to ensure deterministic and consistent outputs.

Furthermore, we evaluate 4 agent frameworks: (1) Function Calling. Models directly use their native tool-use capabilities to solve user queries, without additional prompting. (2) Reminder (Zhao et al., 2025). Models are explicitly reminded in the prompt to consider the user's expressed preferences from the interaction history. (3) Reflexion (Shinn et al., 2023). Models first generate an initial solution for the query. If the solution fails, the model reflects on the error and generates a revised trajectory. (4) RAG (top-3). For each query, an embedding model (Qwen3-Embedding-0.6B (Zhang et al., 2025)) retrieves the 3 interaction sessions most relevant to the query from the interaction history. The retrieved sessions are then appended to form the new interaction history given to LLMs. The prompts used for the agent frameworks are provided in Appendix A.5. By default, we adopt the Function Calling setting when evaluating LLMs, unless otherwise specified.

3.2 Performance of Models on APOLLO

In this section, we evaluate the performance of different models on APOLLO, with results summarized in Table 1. From the table, we make the following observations:

(1) LLMs and LRMs still face challenges in following user preferences. For example, even state-of-the-art models such as GPT-4o and GPT-OSS-120B achieve accuracies of only 0.4827 and 0.2755, respectively, when the context length reaches $12k$. Similarly, Gemini-2.5-Flash, which demonstrates strong reasoning ability in domains such as mathematics and code, achieves only 0.3928 accuracy and 0.6089 F1 score when the context length is $2k$. These results suggest that current models still struggle to reliably extract user preferences from the interaction history and adhere to them when calling tools.

(2) Longer interaction histories degrade preference following performance. For GPT-4o-mini, the accuracies at context lengths of $2k$, $4k$, $8k$, and $12k$ are 0.4764, 0.3971, 0.3678, and 0.3663, respectively. A possible explanation is that as the interaction history grows longer, the burden on models to process and interpret the context increases, making it more difficult to accurately locate and recall the user preferences relevant to the query, which in turn hinders preference following during tool use.

(3) Explicit reasoning improves preference following. For Qwen3-8B and Qwen3-8B (think), the average accuracies across different interaction history lengths are 0.3396 and 0.4373, respectively. The benefits of explicit reasoning become even more pronounced with longer contexts. For instance, at $2k$ context length, Qwen3-8B (think) outperforms Qwen3-8B by 0.0298 in accuracy, while at $12k$, the improvement increases to 0.1154. This indicates that explicitly generating reasoning content helps models to recall and identify user preferences in long interaction histories, thereby enhancing preference following in extended contexts.

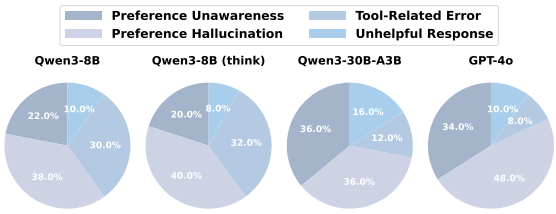

Figure 3: Distribution of 4 error types across different models (evaluated in a context length of $8k$ tokens).

4 Analysis

4.1 Error Analysis

In this section, we conduct an error analysis to better understand why models fail on our benchmark. We categorize the errors into four types: **Preference unawareness**: The model fails to recognize that user preferences expressed in the interaction history should be considered when solving the query. Instead, it either claims that preference information is missing or repeatedly asks the user to provide their preferences. **Preference hallucination**: The model fabricates or hallucinates user preferences when generating tool parameters, rather than adhering to the actual preferences expressed in the interaction history. **Tool-related error**: The model makes mistakes in tool parameters that are irrelevant to the user's preferences (e.g., date, location), or calls an inappropriate tool unrelated to the user's intent. **Unhelpful response**: The model misinterprets the query, producing irrelevant or uninformative responses that fail to address the user's request. For each model, we randomly sample 50 failed trajectories and manually annotate them, assigning each trajectory to one of the four error categories. The results are shown in Figure 3.

As shown in the figure, we observe the following: (1) Preference hallucination is the most frequent error type. For example, in Qwen3-8B and GPT-4o, preference hallucinations account for 38.0% and 48.0% of the errors, respectively. This indicates that models often fail to correctly identify user preferences and instead fabricate them when calling tools, which could substantially degrade user satisfaction in real-world scenarios. (2) GPT-4o exhibits a notably lower error rate in tool-related errors, which may partly explain why it outperforms other models on APOLLO. This suggests that even under long-context settings, GPT-4o maintains relatively high-quality tool-use behavior. (3) Explicit reasoning does not alter the distribution of error types. For instance, the error distributions of Qwen3-8B and Qwen3-8B (think) are highly similar, indicating that while explicit reasoning improves overall preference following performance, it does not fundamentally change the kinds of errors the model tends to make.

Model	Preference		Query	
	Explicit	Implicit	Reactive	Proactive
Qwen3-8B	0.4745	0.2261	0.3861	0.2930
Qwen3-8B (think)	0.6305	0.2748	0.4731	0.4014
Qwen3-30B-A3B	0.4350	0.2142	0.3346	0.2954
GPT-OSS-120B	0.5042	0.1814	0.3810	0.2767
GPT-4o	0.7674	0.2998	0.5541	0.4726

Table 2: Model accuracy on different types of preferences and queries. Results are averaged over all context lengths.

4.2 Analysis of Preference and Query Types

In APOLLO, users can express their preferences either through explicit statements or implicit option selections. In addition, user queries fall into two types: reactive and proactive queries. In this section, we analyze how LLMs perform across different preference and query types, and report their accuracies accordingly. Experimental results are demonstrated in Table 2.

From Table 2, we observe that models perform worse on implicit preferences compared to explicit preferences. For example, GPT-OSS-120B achieves an accuracy of 0.5042 on explicit preferences, but only 0.1814 on implicit preferences. This suggests that implicit preferences increase the difficulty and complexity of inferring user intent, making it harder for models to follow user preferences when solving queries. Regarding query types, models generally perform worse on proactive queries than on reactive queries. This is because, in proactive queries, users do not explicitly state their needs, requiring the model to infer them from the surrounding environmental context, which further challenges its ability to ground the query in the correct user preference. Interestingly, we observe that GPT-4o achieves comparable performance across reactive and proactive queries (0.5541 and 0.4726, respectively), indicating its stronger ability to correctly infer user intent from context. Given the importance of enabling LLM-based agents to act proactively through accurate understanding of environmental signals, future research should place greater emphasis on improving models' proactive reasoning and preference-grounding capabilities.

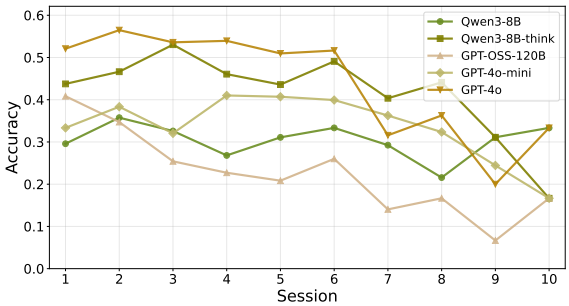

Figure 4: The accuracy of models when user preferences appear in different sessions within the interaction history. The interaction history is fixed to a total of 10 sessions.

4.3 Analysis of Preference Position

In our benchmark, user preference-related sessions are randomly positioned within the interaction history. To analyze the impact of preference position on preference following, we select samples whose interaction histories contain 10 sessions and compute model accuracy with respect to the position of the preference-related session. As shown in Figure 4, we observe that model accuracy decreases as the preference-related session appears later in the interaction history. For example, for GPT-4o, when the preference-related session occurs at the 3rd, 6th, and 9th positions, the corresponding accuracies are 0.5359, 0.5165, and 0.2000, respectively. A possible explanation is that when the preference appears closer to the end of the interaction history, it must compete with the user query for the model's attention. Since LLMs tend to allocate substantial attention to understanding and parsing the query, their focus on retrieving the relevant preference diminishes, thereby reducing their ability to adhere to user preferences (Li et al., 2023b; Yu et al., 2024).

5 Impact of Agent Frameworks

In this section, we investigate the impact of different agent frameworks, Reminder, Reflexion, and RAG, on the ability of LLMs to follow user preferences. As shown in Figure 5, all three frameworks outperform the native Function Calling baseline, with Reflexion achieving the most substantial gains. For instance, on Qwen3-8B, Reflexion improves accuracy by 0.0466 and 0.1038 at context lengths of $2k$ and $8k$, respectively. This suggests that LLMs can effectively reflect on the causes of trajectory failures and leverage user preferences encoded in the interaction history to address queries.

For Reminder, while it demonstrates promising results in dialogue scenarios, its improvements on APOLLO are limited compared to Function Calling. Specifically, on Qwen3-8B, Reminder improves average accuracy by only 0.0144 over Function Calling, whereas Reflexion achieves an average gain of 0.0743. A plausible explanation is that in APOLLO, user preferences and query formulations are highly diverse, making it challenging for models to reliably identify preferences from the interaction history, even when they are explicitly highlighted in the prompt.

For RAG, the framework retrieves a subset of the interaction history most relevant to the query and feeds it into the model, thereby alleviating the difficulty of extracting preferences under long contexts. For example, when the context length ranges from $4k$ to $12k$, RAG maintains stable accuracy, highlighting its potential to enable preference following in longer interaction histories.

6 Improving Preference Following in LLMs with RL

To enhance the preference following ability of LLMs, we train Qwen3-8B using RL in this section. First, we partition the APOLLO users into training (15 users) and test (5 users) sets to prevent potential preference leakage. Next, we select $8k$ context length samples from the training users to form the training data. For evaluation, we use samples of all context lengths from the test users, which allows us to assess the generalization of the RL-trained model across different context lengths. We train Qwen3-8B using the GRPO algorithm (Guo et al., 2025) on the training data, where the model is encouraged to first output its reasoning process before performing the corresponding tool calls, which helps it better adhere to user preferences (see Section 3.2). For reward design, to teach the model to follow user preferences correctly when calling tools, we adopt accuracy as the rule-based reward function. Implementation details of the RL process are provided in Appendix A.2.

Figure 6 shows the accuracy of different models on the test set. We observe that: (1) RL training significantly enhances the model's preference following ability. For example, under the Function Calling framework with a context length of $8k$, Qwen3-8B (think) achieves an accuracy of 0.4516, whereas Qwen3-8B (think)-RL reaches 0.7248. This demonstrates that RL training effectively encourages the model to optimize toward maximizing the reward, thereby improving accu-

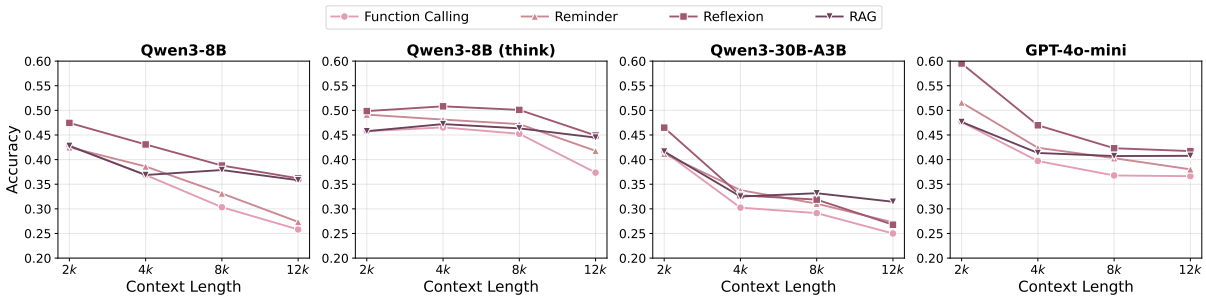

Figure 5: Performance comparison of different agent frameworks on APOLLO.

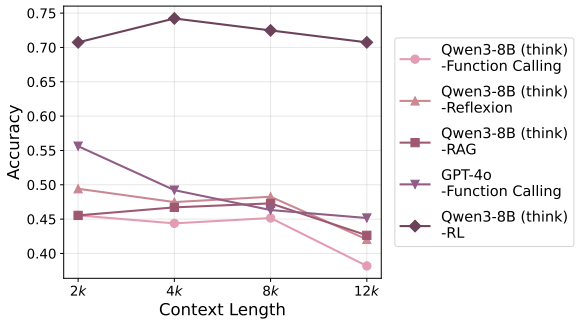

Figure 6: Performance of the RL-trained model and baseline models on 5 unseen users.

racy. (2) The RL-trained model exhibits generalization across context lengths. Although it is trained only on samples with a context length of $8k$, it still achieves substantial gains on other lengths; for instance, Qwen3-8B (think)-RL reaches an accuracy of 0.7074 with a $12k$ context length. Overall, the results confirm the effectiveness of RL in improving the model's ability to follow user preferences.

7 Related Work

Tool use in LLMs. To comprehensively evaluate the tool-use capabilities of LLMs, several benchmarks have been proposed to assess their performance across dimensions such as accuracy (Qin et al., 2023; Li et al., 2023a), safety (Ye et al., 2024; Chen et al., 2024b), and efficiency (Huang et al., 2023). To further enhance the abilities of open-source models in these dimensions, researchers have explored collecting high-quality tool-use trajectories from stronger LLMs and fine-tuning models on the trajectories (Liu et al., 2024a,b), or designing reward functions to optimize models via RL (Feng et al., 2025; Jin et al., 2025). However, existing efforts on tool use primarily focus on improving task performance while largely overlooking the need to account for personalized user preferences, where the same task may lead to different tool-use trajectories depending on the user (Hao et al., 2025; Kim et al., 2025). In this work, we introduce APOLLO, a benchmark specifically designed to evaluate the ability of LLMs to follow personalized user preferences in tool-use scenarios.

Preference following in LLMs. Preference following requires LLMs to proactively learn and adapt to a user's personalized background and requirements, and to adjust their behavior accordingly (Chen et al., 2024a; Jiang et al., 2024; Liu et al., 2025; Jiang et al., 2025). This capability plays a critical role in applications such as recommendation systems (Wu et al., 2024a; Wang et al., 2025), dialogue systems (Hong et al., 2024), and role-playing agents (Shanahan et al., 2023; Ge et al., 2024). To align LLMs with user preferences in tool-use scenarios, existing work typically either explicitly provides preference information as input to the model (Hao et al., 2025), or allows the model to obtain preferences directly through tool calling (Cheng et al., 2025). In this work, we propose a more realistic benchmark, where user preferences are not explicitly provided but must instead be inferred from interaction histories (Salemi et al., 2023; Zhao et al., 2025; Jiang et al., 2025). LLMs are then required to execute actions based on the inferred preferences.

8 Conclusions

In this work, we introduce APOLLO, a benchmark designed to evaluate the ability of LLMs to follow user preferences in tool calling scenarios. Specifically, APOLLO considers both explicit and implicit user preferences as well as reactive and proactive queries, posing challenges for LLMs to correctly identify and adhere to user preferences. Based on APOLLO, we conduct extensive experiments to assess the performance of various models under different agent frameworks. Finally, we demonstrate that RL effectively enhances the model's capability to follow user preferences.

Limitations

In this work, we propose a benchmark to evaluate the preference following ability of LLMs in tool calling scenarios. In our benchmark, the maximum context length is $12k$ tokens. However, real-world user-agent interaction histories can be much longer, sometimes reaching up to millions of tokens. Future work should explore how well models can maintain preference adherence in such long-context settings. In addition, our benchmark currently covers only 4 domains. In practice, users also exhibit personalized preferences in a wider range of scenarios, such as shopping and healthcare. Incorporating these domains would allow for a more comprehensive evaluation of models' preference following capabilities across diverse contexts.

References

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, and 1 others. 2025. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*.

Jin Chen, Zheng Liu, Xu Huang, Chenwang Wu, Qi Liu, Gangwei Jiang, Yuanhao Pu, Yuxuan Lei, Xiaolong Chen, Xingmei Wang, and 1 others. 2024a. When large language models meet personalization: Perspectives of challenges and opportunities. *World Wide Web*, 27(4):42.

Zhi-Yuan Chen, Shiqi Shen, Guangyao Shen, Gong Zhi, Xu Chen, and Yankai Lin. 2024b. Towards tool use alignment of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1382–1400.

Zihao Cheng, Hongru Wang, Zeming Liu, Yuhang Guo, Yuanfang Guo, Yunhong Wang, and Haifeng Wang. 2025. Toolspectrum: Towards personalized tool utilization for large language models. *arXiv preprint arXiv:2505.13176*.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. 2025. Retool: Reinforcement learning for strategic tool use in llms. *arXiv preprint arXiv:2504.11536*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023a. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. 2023b. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1).

Tao Ge, Xin Chan, Xiaoyang Wang, Dian Yu, Haitao Mi, and Dong Yu. 2024. Scaling synthetic data creation with 1,000,000,000 personas. *arXiv preprint arXiv:2406.20094*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Yupu Hao, Pengfei Cao, Zhuoran Jin, Huanxuan Liao, Yubo Chen, Kang Liu, and Jun Zhao. 2025. Evaluating personalized tool-augmented llms from the perspectives of personalization and proactivity. *arXiv preprint arXiv:2503.00771*.

Mengze Hong, Chen Jason Zhang, Chaotao Chen, Rongzhong Lian, and Di Jiang. 2024. Dialogue language model with large-scale persona data engineering. *arXiv preprint arXiv:2412.09034*.

Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Zhenqiang Gong, and 1 others. 2023. Metatool benchmark for large language models: Deciding whether to use tools and which to use. *arXiv preprint arXiv:2310.03128*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Bowen Jiang, Zhuoqun Hao, Young-Min Cho, Bryan Li, Yuan Yuan, Sihao Chen, Lyle Ungar, Camillo J Taylor, and Dan Roth. 2025. Know me, respond to me: Benchmarking llms for dynamic user profiling and personalized responses at scale. *arXiv preprint arXiv:2504.14225*.

Ruili Jiang, Kehai Chen, Xuefeng Bai, Zhixuan He, Juntao Li, Muyun Yang, Tiejun Zhao, Liqiang Nie, and Min Zhang. 2024. A survey on human preference learning for large language models. *arXiv preprint arXiv:2406.11191*.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*.

Tae Soo Kim, Yoonjoo Lee, Yoonah Park, Jiho Kim, Young-Ho Kim, and Juho Kim. 2025. Cupid: Evaluating personalized and contextualized alignment of llms from interactions. *arXiv preprint arXiv:2508.01674*.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023a. Api-bank: A comprehensive benchmark for tool-augmented llms. *arXiv preprint arXiv:2304.08244*.

Zongjie Li, Chaozheng Wang, Pingchuan Ma, Daoyuan Wu, Shuai Wang, Cuiyun Gao, and Yang Liu. 2023b. Split and merge: Aligning position biases in large language model based evaluators. *arXiv preprint arXiv:2310.01432*.

Jiahong Liu, Zexuan Qiu, Zhongyang Li, Quanyu Dai, Jieming Zhu, Minda Hu, Menglin Yang, and Irwin King. 2025. A survey of personalized large language models: Progress and future directions. *arXiv preprint arXiv:2502.11528*.

Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, and 1 others. 2024a. Toolace: Winning the points of llm function calling. *arXiv preprint arXiv:2409.00920*.

Zuxin Liu, Thai Hoang, Jianguo Zhang, Ming Zhu, Tian Lan, Juntao Tan, Weiran Yao, Zhiwei Liu, Yihao Feng, Rithesh RN, and 1 others. 2024b. Apigen: Automated pipeline for generating verifiable and diverse function-calling datasets. *Advances in Neural Information Processing Systems*, 37:54463–54482.

Yaxi Lu, Shenzhi Yang, Cheng Qian, Guirong Chen, Qinyu Luo, Yesai Wu, Huadong Wang, Xin Cong, Zhong Zhang, Yankai Lin, and 1 others. 2024. Proactive agent: Shifting llm agents from reactive responses to active assistance. *arXiv preprint arXiv:2410.12361*.

Zile Qiao, Guoxin Chen, Xuanzhong Chen, Donglei Yu, Wenbiao Yin, Xinyu Wang, Zhen Zhang, Baixuan Li, Huifeng Yin, Kuan Li, and 1 others. 2025. Webresearcher: Unleashing unbounded reasoning capability in long-horizon agents. *arXiv preprint arXiv:2509.13309*.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, and 1 others. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. 2023. Lamp: When large language models meet personalization. *arXiv preprint arXiv:2304.11406*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.

Murray Shanahan, Kyle McDonell, and Laria Reynolds. 2023. Role play with large language models. *Nature*, 623(7987):493–498.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36:38154–38180.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.

Qwen Team. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2:3.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, and 1 others. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.

Lei Wang, Jingsen Zhang, Hao Yang, Zhi-Yuan Chen, Jiakai Tang, Zeyu Zhang, Xu Chen, Yankai Lin, Hao Sun, Ruihua Song, and 1 others. 2025. User behavior simulation with large language model-based agents. *ACM Transactions on Information Systems*, 43(2):1–37.

Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, and 1 others. 2024a. A survey on large language models for recommendation. *World Wide Web*, 27(5):60.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, and 1 others. 2024b. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First Conference on Language Modeling*.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, and 1 others. 2025. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Bufang Yang, Lilin Xu, Liekang Zeng, Kaiwei Liu, Siyang Jiang, Wenrui Lu, Hongkai Chen, Xiaofan Jiang, Guoliang Xing, and Zhenyu Yan. 2025b. Contextagent: Context-aware proactive llm agents with open-world sensory perceptions. *arXiv preprint arXiv:2505.14668*.

Junjie Ye, Sixian Li, Guanyu Li, Caishuang Huang, Songyang Gao, Yilong Wu, Qi Zhang, Tao Gui, and Xuanjing Huang. 2024. Toolsword: Unveiling safety issues of large language models in tool learning across three stages. *arXiv preprint arXiv:2402.10753*.

Yijiong Yu, Huiqiang Jiang, Xufang Luo, Qianhui Wu, Chin-Yew Lin, Dongsheng Li, Yuqing Yang, Yongfeng Huang, and Lili Qiu. 2024. Mitigate position bias in large language models via scaling a single dimension. *arXiv preprint arXiv:2406.02536*.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, and 1 others. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

Siyan Zhao, Mingyi Hong, Yang Liu, Devamanyu Hazarika, and Kaixiang Lin. 2025. Do llms recognize your preferences? evaluating personalized preference following in llms. *arXiv preprint arXiv:2502.09597*.

A Appendix

A.1 Domain and Tools

We present the domains and corresponding tools in our benchmark in Table 3. In the table, the commute, travel, entertainment, and food domains are related to user preferences, whereas the remaining domains are employed to generate unrelated sessions.

A.2 Implementation Details for RL Process

Our RL training is implemented based on the VeRL training library (Sheng et al., 2024). We train Qwen3-8B (Yang et al., 2025a) using the GRPO algorithm (Guo et al., 2025) for 50 steps, with a batch size of 256 and a rollout number of 8. We adopt the AdamW optimizer with a learning rate of $1e-6$. For reward design, to teach the model to follow user preferences correctly when calling tools, we adopt a rule-based reward function based on accuracy, defined as

$$Reward = \begin{cases} 1, & \text{if } T_t = T_g \\ 0, & \text{otherwise} \end{cases}$$

where T_t and T_g denote the tool calling sequences produced by the model and the gold reference, respectively. All experiments are conducted on 8 NVIDIA A100 GPUs.

A.3 Preferences and Queries Examples

In our benchmark, user preferences are expressed in two forms: explicitly through direct statements and implicitly through option selections. User queries fall into two categories: reactive and proactive. Examples of user preferences and queries are provided in Table 4 and Table 5, respectively.

A.4 Prompt for Preference-Related Session Generation

In Table 6, we present the prompt used with GPT-4.1 to generate preference-related sessions.

A.5 Prompts for Agent Frameworks

In this work, we evaluate the impact of different agent frameworks, Function Calling, Reminder, Reflexion, and RAG, on the preference following capability of LLMs. The detailed prompts for these frameworks are presented in Table 7. For Function Calling, we rely on the model's native tool-calling ability to solve user queries. For Reminder, we augment the prompt with an explicit reminder encouraging the model to consider user preferences mentioned in the interaction history that are relevant to the current query. For Reflexion, the model is instructed to reflect on previous failed solutions and generate a revised tool-calling trajectory. For RAG, the only difference from Function Calling is that it retrieves the 3 most relevant sessions from the interaction history as the new context; the prompt remains the same, so we omit it from the table.

Domain	Tool
Commute	get_bus_eta, get_weather, get_distance, request_ride, transport_recommendation
Travel	search_flight, book_flight, search_hotel, book_hotel
Entertainment	search_movie, book_movie_ticket, search_music
Food	search_beverage_id, order_beverage, order_food_delivery
Communication	email_assistant, translation_tool, push_notification, get_latest_messages
Games	game_recommendation, player_stats_query, multiplayer_matchmaking
Education	course_search, homework_checker, exam_practice, vocabulary_trainer
Finance	get_stock_info, loan_repayment_calculator, currency_converter, investment_return_calculator, get_loan_interest_rates
Health	health_advice, sleep_duration, exercise_recommendation, daily_water_intake, sleep_quality_checker, recipe_search, nutrition_analysis
News	search_news, get_top_headlines, get_news_summary
Shopping	search_products, get_trending_products, check_product_availability, get_product_details, subway_route_search

Table 3: Domains and their corresponding tools in APOLLO.

Explicit Preference	Implicit Preference
User: I'm traveling from Orlando to Seattle today for my work with a non-profit organization. I value reliability and comfort during my travels. For flights, I prefer direct flights to minimize travel time, and I choose economy cabin for a balance between comfort and cost. I prefer to pay for flights with a credit card and always purchase flight insurance. Please help me book my flight for today. **Assistant**: Understood. I'll begin by searching for available flights from Orlando to Seattle for today.	**User**: Hi, I'm traveling from Orlando to Seattle today for my work with a non-profit organization. I need to book a flight and a hotel. Can you help me check available flights first? **Assistant**: Sure, I'll search for available flights from Orlando to Seattle today. **Assistant**: Function(name="search_flight", arguments={"date": "2025-05-02", "origin": "Orlando", "destination": "Seattle"}) **Tool**: ["flight_id": "FL6522", "flight_type": "connecting", "cabin": "basic_economy", "flight_id": "FL6630", "flight_type": "connecting", "cabin": "economy", "flight_id": "FL3036", "flight_type": "connecting", "cabin": "business", "flight_id": "FL8849", "flight_type": "direct", "cabin": "basic_economy", "flight_id": "FL2900", "flight_type": "direct", "cabin": "economy", "flight_id": "FL9154", "flight_type": "direct", "cabin": "business"] **Assistant**: Here are the available flights: - Connecting: FL6522 (basic economy), FL6630 (economy), FL3036 (business) - Direct: FL8849 (basic economy), FL2900 (economy), FL9154 (business) Which type of flight would you like to book: connecting or direct? And which cabin do you prefer: basic economy, economy, or business? **User**: I prefer direct flights to save time, and I'd like to book an economy seat. Please help me book flight FL2900. **Assistant**: How would you like to pay for your flight ticket? The options are credit card, PayPal, or Apple Pay. Also, would you like to purchase travel insurance for your flight? **User**: I'll pay with my credit card, and yes, I want to buy flight insurance.
User: I'd like to watch a movie tonight. I prefer dramas and enjoy going to Cinemark Theatres because their seats are comfortable and the environment is quiet. I prefer the classic experience, so I usually book tickets for the standard hall. I also like to pay with my credit card to get reward points. Please help me search for a suitable drama movie and book a ticket for me. **Assistant**: Got it. I'll start by searching for drama movies showing tonight at Cinemark Theatres.	**User**: I'm thinking about watching a movie today. Can you help me find something suitable? **Assistant**: Of course! Do you have a preferred movie genre, such as action, drama, or comedy? Also, which cinema would you like to go to? Options include Cinemark Theatres, AMC Theatres, and Regal Cinemas, among others. **User**: I enjoy drama movies and usually go to Cinemark Theatres for a comfortable and quiet experience. **Assistant**: Great! I'll search for drama movies playing at Cinemark Theatres today.
User: I am in Orlando today and I have a busy day bridging health and social services. I like to order coffee from Tim Hortons because I find its taste comforting during work. I usually prefer my coffee at room temperature so I can sip it slowly while working. I also always ask for less sugar to keep my beverage healthier and not too sweet. **Assistant**: Sure, I will order a coffee for you based on your preferences. **Assistant**: Function(name="search_beverage_id", arguments="date": "2025-05-12", "city": "Orlando")	**User**: Hi, I have a long day ahead and I'd like to order a beverage to keep me energized. Assistant: Of course! Would you like coffee, tea, juice, or something else? **User**: I'd like coffee, please. It's comforting for me during busy workdays. **Assistant**: Great choice. Do you have a preferred coffee brand, such as Tim Hortons, Costa Coffee, or Luckin Coffee? **User**: Tim Hortons is my usual pick. **Assistant**: Noted. How would you like your coffee served? Regular ice, less ice, or room temperature? **User**: Room temperature works best for me-I like to sip it slowly while working. **Assistant**: And what about the sugar level? Would you prefer regular, less sugar, or no sugar? **User**: Less sugar, please. I'm trying to keep it a bit healthier. **Assistant**: Got it. I'll order a Tim Hortons coffee at room temperature with less sugar for you.

Table 4: Examples of user preferences.

Reactive Query	**Proactive Query**
I am at home right now and would like to visit 1500 Orange Avenue for a coffee break. Can you suggest how I should get there?	Before leaving her home, Alice mentions to her roommate that she is going to meet a colleague at 100 International Drive.
I'm planning my commute from my house to my workplace. Can you suggest the best transportation option for me?	Alice checks her watch and says to her partner that she needs to get to the office.
I am currently in Orlando and need to travel to Washington tomorrow for my work with a non-profit organization focused on bridging health and social services. Could you assist me in finding and purchasing a suitable flight ticket?	The user's calendar notifies her to arrange her flight from Orlando to Seattle tomorrow for her non-profit work supporting health and social services.
Can you assist me in booking a movie ticket for tomorrow night?	Alice texts her sibling asking if they want to join her for a movie tomorrow, mentioning she is about to book a ticket for herself.
I'm at my company right now. Could you help me place an order for food delivery?	The calendar shows that it's lunchtime and the user's phone is located at her company. The user opens a food delivery app on her computer.

Table 5: Examples of user queries.

Below are the user's profile, preference, and the tools available for use.
User profile:
{user profile}

User preferences:
{user preference}

Tool definitions and output examples:
{tool information}
You need to generate dialogues between the user and the assistant that fully reflect the user's preferences.
- Your generated dialogues should related to user profile.
- The assistant is not aware of any user preferences or information unless explicitly provided by the user, so avoid having the assistant mention or reference any user preferences in its responses.
- The assistant can make use of tools, but is restricted to calling only one tool at a time. Furthermore, in each turn, the assistant may either engage in dialogue or make a tool call, but not both. You should also refer to the tool output examples to simulate the tool's output.
- The user should have the assistant call tools that are **only** relevant to their preferences and avoid unnecessary tool calls.

Below are dialogue examples:
{example}

Now, generate dialogues that can fully reflect the user's preferences, following the format of the example.

Table 6: Detailed prompt used to generate preference-related sessions.

Function Calling

System Prompt

You are a helpful agent. Respond to the user request according to the following policies:
- You should only make one tool call at a time. Furthermore, in each turn, you may either make a tool call or output response content, but not both.
- You only need to respond to the user's request, without asking the user any additional questions.
- If you believe the user's request has been fully addressed, generate [Finish] as a standalone message without anything else to end the response.

User Prompt

<User Information>
{user profile}
<Conversation History>
{interaction history}
<Request>
{query}

Reminder

System Prompt

You are a helpful agent. Respond to the user request according to the following policies:
- You should only make one tool call at a time. Furthermore, in each turn, you may either make a tool call or output response content, but not both.
- You only need to respond to the user's request, without asking the user any additional questions.
- If you believe the user's request has been fully addressed, generate [Finish] as a standalone message without anything else to end the response.

As a personalized agent, please consider the preferences related to the request that are revealed in the user's conversation history, and ensure that your responses adhere to these preferences.

User Prompt

<User Information>
{user profile}
<Conversation History>
{interaction history}
<Request>
{query}

Reflexion

System Prompt

You are a helpful agent. Respond to the user request according to the following policies:
- You should only make one tool call at a time. Furthermore, in each turn, you may either make a tool call or output response content, but not both.
- You only need to respond to the user's request, without asking the user any additional questions.
- If you believe the user's request has been fully addressed, generate [Finish] as a standalone message without anything else to end the response.

In addition, I will provide you with a previously failed solution. Based on this failed solution, please reflect on the reasons for its failure and respond to the user's request again.

User Prompt

<User Information>
{user profile}
<Conversation History>
{interaction history}
<Request>
{query}
Here is the previously failed solution:
<Failed Solution>
{previous solution}
Now, respond to the user's request again.

Table 7: Prompts for different agent frameworks. The RAG framework adopts the same prompt as the Function Calling setup.