

FAMA: Failure-Aware Meta-Agent Framework for Open-Source LLMs in Interactive Tool Use Environments

Amir Saeidi^{†*} Venkatesh Mishra^{†*} Souradeep Mukhopadhyay[†]
Gaowen Liu[‡] Ali Payani[‡] Jayanth Srinivasa[‡] Chitta Baral[†]

[†]Arizona State University [‡]Cisco Research

{saeidi1, vmishr23, chitta}@asu.edu, {gaoliu, apayani, jasriniv}@cisco.com

Abstract

Large Language Models are being increasingly deployed as the decision-making core of autonomous agents capable of effecting change in external environments. Yet, in conversational benchmarks, which simulate real-world customer-centric issue resolution scenarios, these agents frequently fail due to the cascading effects of incorrect decision-making. These challenges are particularly pronounced for open-source LLMs with smaller parameter sizes, limited context windows, and constrained inference budgets, which contribute to increased error accumulation in agentic settings. To tackle these challenges, we present the **Failure-Aware Meta-Agent (FAMA)** framework. FAMA operates in two stages: first, it analyzes failure trajectories from baseline agents to identify the most prevalent errors; second, it employs an orchestration mechanism that activates a minimal subset of specialized agents tailored to address these failures by injecting a targeted context for the tool-use agent before the decision-making step. Experiments across open-source LLMs demonstrate performance gains up to **27%** across evaluation modes over standard baselines. These results highlight that targeted curation of context through specialized agents to address common failures is a valuable design principle for building reliable, multi-turn tool-use LLM agents that simulate real-world conversational scenarios.

1 Introduction

Recent advancements in large language models (LLMs) (Liu et al., 2025a; Anthropic, 2025; Google DeepMind, 2025; OpenAI, 2025) have significantly enhanced their potential to function as the decision-making backbone of autonomous agents (Chow et al., 2025). Such agents are now being tested across a broad spectrum of applications, ranging from customer support and enterprise automation

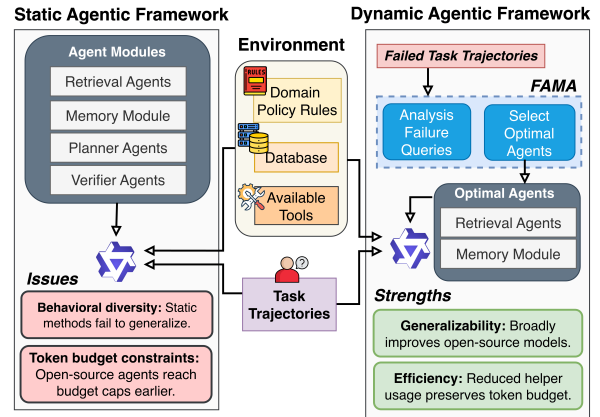


Figure 1: Comparison of static and dynamic (e.g., FAMA) agentic orchestration.

(Kemell et al., 2025; Awad et al., 2025; Almeida, 2025) to scientific discovery (Zhou et al., 2025; Kumbhar et al., 2025), software engineering (Dong et al., 2025), and data analysis (Rahman et al., 2025), where these systems must reason over long conversational horizons and generate tokens which form the basis of decisions and actions with real effects on external environments. One of the most important use-cases of these agents is of multi-turn conversational, interactive chatbots, which have augmented external API (also termed as a ‘tool’) invocation capabilities (Wang et al., 2025a; Zhang et al., 2025a; Farn and Shin, 2023). Despite impressive progress, recent benchmarks such as τ -bench (Yao et al., 2024), τ -trait (He et al., 2025) and ACEBench (Chen et al., 2025a) reveal that even strong LLM-based agents struggle to reliably and efficiently complete realistic, customer-centric tasks that require sustained reasoning and structured tool use.

A central challenge exposed by tool-use benchmarks is the accumulation of decision-making errors over long trajectories (Sinha et al., 2025; Mishra et al., 2025; Lichkovski et al., 2025) that involve heterogeneous context, including tool out-

* Equal Contribution

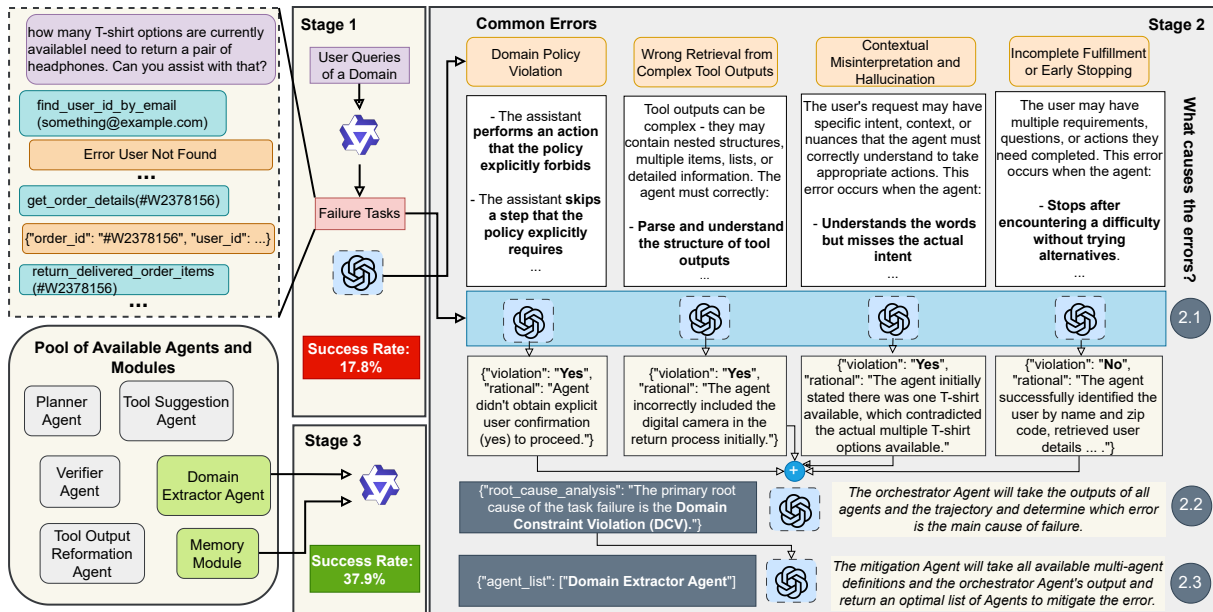


Figure 2: Overview of the FAMA framework. **Stage 1:** A baseline agent, without multi-agent assistance, is executed on domain tasks. **Stage 2:** failed tasks are analyzed and categorized by a human or an agentic framework. **Stage 2.1:** Independent agents focus on distinct issues and analyze the causes of failure through task-specific conversations. **Stage 2.2:** An orchestrator agent identifies the primary reasons for failure. **Stage 2.3:** A mitigation agent selects the most appropriate helper agents to address the identified issues. **Stage 3:** the baseline agent is re-executed using only the helper agents recommended by the mitigation agent.

puts, domain rules, inter-agent interventions, ambiguous user intent and evolving task constraints. (Sinha et al., 2025; Handa et al., 2025a; Lichkovski et al., 2025). Our experiments (§5.2) show that such errors are heterogeneous: different model backbones, particularly open-source models (Yang et al., 2025; Abdin et al., 2024) with limited context windows, exhibit distinct dominant failure modes, indicating that static prompting strategies or agentic scaffolding architectures are insufficient.

Prior work has largely sought to improve tool-using agents through task-specific supervised fine-tuning (Chen et al., 2023; Zeng et al., 2024), or reinforcement learning-based optimization (Acikgoz et al., 2025; Zhao et al., 2025b; Jiang et al., 2025). For multi-turn tool-use/tool-calling tasks, where trajectories are lengthy, partially observable, and highly variable (Kate et al., 2025; Wang et al., 2025b): collecting sufficient high-quality supervision or reward-aligned experience becomes prohibitively expensive (Saeidi et al., 2026; Golubev et al., 2025; Qi et al., 2025; Chen et al., 2025b). Reinforcement learning methods (Zhang et al., 2025b) are not overall effective for these challenges, as training requires large-scale curated trajectories, repeated execution of complex tool interactions and long episodes to propagate sparse and delayed re-

wards, leading to substantial compute and engineering overhead (Wang et al., 2023, 2025c). Moreover, because training objectives are typically optimized over learning patterns of correct behavior, these methods rarely target the specific failure patterns exhibited by a given agent (Rauba et al., 2024), limiting their effectiveness in addressing systematic errors observed in practice.

We have also observed that there has been limited exploration (Nekoei et al., 2025; Fu et al., 2024) of training-free improvement techniques that extract failure signals from agent trajectories and use them to modularly orchestrate agentic systems built specifically on smaller open-source language models (Belcak et al., 2025). While larger models benefit from extensive pretraining and long context windows, their deployment is often impractical in real-world settings due to cost, latency, and privacy constraints (Liang and Tong, 2025). Moreover, scaling alone can mask, rather than resolve, fundamental deficiencies in long-horizon decision-making and tool use. Small language models expose these limitations more clearly (Shen et al., 2024; Sinha et al., 2025), making them an ideal testbed for studying systematic agent failures under realistic resource constraints.

In this work, we argue that reliable multi-turn

tool use requires failure-aware agent composition. We introduce a two-stage framework, named the **Failure-Aware Meta-Agentic (FAMA)** framework, that explicitly identifies the most prevalent failure modes in language agent conversation trajectories and dynamically adapts the tool-use agent’s internal prior context to mitigate those failures. FAMA operates in two stages. First, it analyzes failure trajectories produced by a baseline agent and categorizes errors into a small set of common failure types. *Rather than treating all errors equally, this analysis surfaces the dominant errors that most strongly impact performance.*

In the second stage, the FAMA framework employs a lightweight routing mechanism consisting of an orchestrator and a mitigation agent (Figure 2) that selectively activates a set of specialized LLM helper-agents designed to address these specific weaknesses (see §4.1). By dynamically constructing the agent’s prior context using only the necessary agents, FAMA avoids the overhead of excessive context injection while directly addressing the agent’s most critical shortcomings (Figure 1). This design motivates our use of the term ‘*meta-agentic*’, as the proposed system does not act in the environment directly, but operates at a higher level by reasoning over agent behaviors, diagnosing their failures, and dynamically composing a minimal set of specialized agents to shape the decision-relevant context of the underlying tool-use agent.

We evaluate FAMA across τ -bench, τ -trait and ACEBench using a set of open-source LM backbones with parametric sizes ranging from 4-72 billion. Our results demonstrate consistent improvements in tool-use accuracy, trajectory reliability, and context efficiency. FAMA yields noticeable task-success gains of up to **25%** in τ -bench, **27%** in ACEBench and **24%** in τ -trait over baselines for open-source model backbones, highlighting the effectiveness of targeted error mitigation over brute-force context injection. In summary, our main contributions are:

1. We propose the **Failure-Aware Meta-Agentic (FAMA)** framework, which operates in two stages: first, identifying the predominant errors of baseline agents, and then dynamically constructing the prior context for the tool-use agent by selectively enabling the appropriate agents designed to mitigate these errors.
2. We demonstrate that the FAMA framework’s

optimized context construction significantly improves tool-use performance of diverse open-source LLM agents across the τ -bench, τ -trait and ACEBench benchmarks.

3. We provide a comprehensive analysis of the improvements observed from FAMA over other baseline agents.

2 Related Works

LLM-based Tool-Use Benchmarks While early benchmarks primarily evaluated isolated tool calls in response to single-turn queries (Patil et al., 2024; Li et al., 2023; Qin et al., 2023), recent work has increasingly shifted toward multi-turn, interactive tool-use benchmarks (Yao et al., 2024; Barres et al., 2025; Chen et al., 2025a; He et al., 2025; Lu et al., 2025a; Liu et al., 2024). These benchmarks expose long-horizon reasoning challenges for language agents (Cemri et al., 2025; Sinha et al., 2025) under partial observability, and sustained environment and task constraints, highlighting the need for agent designs that explicitly address failure accumulation over extended interaction trajectories as a valuable direction.

Training-Free Techniques for Tool-Using Agents

A range of techniques, including structured prompting (Yao et al., 2023; Wei et al., 2022), and modular agent orchestration (Mishra et al., 2025; Zhang et al., 2025e; Zhao et al., 2025a), have been developed to improve language agents without updating model parameters. However, these approaches are predominantly evaluated on strong or proprietary LLM backbones, with large context windows and robust reasoning capabilities (Qu et al., 2025; Chowa et al., 2025). This remains relatively underexplored for smaller open-source models remains comparatively, particularly in tool-use settings (Belcak et al., 2025) where limited context, weaker planning, and error accumulation pose significant challenges (Shen et al., 2024), and training techniques are dominantly explored (Jhandi et al., 2025; Zeng et al., 2025; Ma et al., 2025). FAMA attempts to tackle these challenges by specifically focusing on agentic design through failure modes of smaller open-source LLM agents.

Failure-Aware Agentic Orchestration

Recent work has begun to systematically analyze agent failures in multi-agent LLM systems (Cemri et al., 2025; Lu et al., 2025b), uncovering diverse failure causes (Mishra et al., 2025; Sinha et al., 2025)

and proposing taxonomies and annotation datasets to enable more principled debugging and evaluation (Zhu et al., 2025; Banerjee et al., 2025). Other efforts automate failure tracing and attribution (Zhang et al., 2025c; Ge et al., 2025) to localize error sources. Although reflective (Shinn et al., 2023; Madaan et al., 2023) and experiential (Zhang et al., 2025d; Huang et al., 2025) feedback techniques exist, and recent episodic guidance/hinting works (Nekoei et al., 2025; Fu et al., 2024; Wang et al., 2025d) come close in principle, they do not leverage observed failure patterns to dynamically compose model-specific modular agent scaffolds under resource constraints, an angle we explore with our FAMA framework.

3 Problem Statement

For simulating ideal testbeds of tool-use in multi-turn conversations, we utilize tasks from benchmarks such as τ -bench (Yao et al., 2024), τ -trait (He et al., 2025), and ACEBench (Chen et al., 2025a). These benchmarks are specifically designed to assess the decision-making capabilities of language agents in realistic, multi-turn interaction settings. In these setups, both the user and the customer-service assistant are simulated by LLMs, enabling a controlled yet dynamic environment for analyzing interactive behavior. The customer-service agent is the language agent that generates the tokens signifying which tools are to be invoked, while following the specific domain policies. As explained in Shekkizhar et al. (2025), the benchmarks used here do not evaluate standard multi-agent settings (Stone and Veloso, 2000) as the distinct agents simulating different roles, such as the user or the assistant, can have private internal states and competing utilities. Further details on these benchmarks are provided in Appendix B.

4 Method

Although frontier models in agentic frameworks demonstrate strong performance on complex, dynamic tasks, their deployment is often prohibitively expensive. Additionally, integrating these models into private or proprietary products is often impractical due to cost and security constraints. Open-source models, by contrast, offer a viable alternative; their smaller size enables local deployment, making them more cost-effective and better suited for privacy-sensitive applications.

Recent works (Liu et al., 2025b; Rrv et al., 2025;

Handa et al., 2025b) show that open-source models, along with inference-time frameworks built around them, can achieve performance comparable to frontier models on mathematical and reasoning tasks; however, their use within agentic frameworks to solve a complex task remains challenging. Due to their smaller size and the nature of the data on which they are fine-tuned, these models exhibit distinct failure modes and limitations. Additionally, strict context constraints limit their applicability in complex multi-agent frameworks, where multiple agents must collaborate to solve a task, and rapidly fill up the context windows. These factors lead open-source models to exhibit weaker decision-making capabilities compared to frontier models.

To address these challenges, we propose the **Failure-Aware Meta-Agentic (FAMA)** framework, a general and dynamic framework designed to identify failure patterns in the behavior of open-source model agents and determine an effective set of agents that achieves improved performance under resource constraints. In the following sections, we present and analyze the FAMA framework in detail.

4.1 Failure-Aware Meta-Agentic Framework

We assume the availability of a predefined set of agents \mathcal{A} , where each individual agent is denoted by $a \in \mathcal{A}$, and subsets of agents are denoted by $\mathcal{A}' \subseteq \mathcal{A}$. The full agent set \mathcal{A} is adapted from Mishra et al. (2025), which proposes a collection of specialized agents for improving performance in tool-calling environments, and is further extended with additional modules tailored to the benchmarks evaluated in this work. Specifically, \mathcal{A} includes a Domain Constraints Extractor Agent (DCE), a Tool Suggestion Agent (TSA), a Tool Output Reformulator Agent (TOR), a Planner Agent, a Decision Verifier Agent, and a User Context Manager (Memory) module.

We further assume the availability of a predefined set of general error categories representing common agent failures, denoted by \mathcal{E} , where a realized error instance is denoted by $e \in \mathcal{E}$. Based on prior studies (Mishra et al., 2025; Shekkizhar et al., 2025; Kokane et al., 2024) and our own analysis, we categorize failures in tool-calling environments into four classes: (1) Domain Policy Violation, (2) Incorrect Retrieval from Complex Tool Outputs, (3) Contextual Misinterpretation and Hallucination, and (4) Incomplete Fulfillment or Early Stopping. Detailed definitions of these error categories are

provided in Appendix E.

Given the agent set \mathcal{A} and the error category set \mathcal{E} , we define $|\mathcal{E}|$ independent error analysis agents, each responsible for detecting the cause of failure associated with a specific error category e . These agents operate over a predefined list of failure causes derived from empirical analysis of baseline trajectories. Additional details on the causes of failure for each error category are provided in Appendix E. Each error analysis agent produces both a categorical decision and a supporting rationale in textual form.

The textual outputs and rationales produced by all error analysis agents are concatenated into a single input and, together with the full interaction trajectory between the user and the tool-calling agent for a given task τ , are passed to the orchestrator agent for final failure attribution. This process explicitly accounts for cases in which an agent may initially make incorrect decisions but later recover after receiving feedback from the environment, ensuring that failure attribution reflects the overall trajectory rather than isolated errors.

In the final stage, conditioned on the error categories identified by the orchestrator agent and the functional definitions of agents in \mathcal{A} , a dedicated mitigation agent determines an efficient subset of agents $\mathcal{A}^* \subseteq \mathcal{A}$ that can mitigate a specific error category e . Aggregating the mitigation agent’s recommendations across the selected error categories yields a final agent subset that represents the minimal configuration required to address the dominant failure modes.

We collect these recommendations across tasks and deploy multi-agent frameworks composed only of the selected agent subsets (See Algorithm 1). The results reported in Tables 1-4 demonstrate that FAMA consistently achieves better performance across different settings compared to IRMA, ReAct, and FC on both τ -bench, τ -trait, and ACEBench across different models.

5 Experiments

5.1 Experimental Setup

In this section, we mention the models and benchmarks used in our experiments. Our primary focus is on open-source models; accordingly, we evaluate their performance exclusively on tool-calling benchmarks.

Models. We evaluate models from the Qwen family, including **Qwen3-4B-Instruct**, **Qwen3-14B**,

Algorithm 1 Failure-Aware Meta-Agent (FAMA)

Require: Agent set \mathcal{A} ; error categories \mathcal{E} ; task set $\mathcal{T} = \{\tau_1, \dots, \tau_m\}$

Ensure: Selected agent subsets $\{\mathcal{A}_\tau^*\}_{\tau \in \mathcal{T}}$, where $\mathcal{A}_\tau^* \subseteq \mathcal{A}$

```

1: Initialize failure set  $\mathcal{F} \leftarrow \emptyset$ 
2: for each task  $\tau \in \mathcal{T}$  do
3:    $(\xi_\tau, r_\tau) \leftarrow \text{EXECUTE}(\tau)$   $\triangleright \xi_\tau$ : full
   interaction trajectory
4:   if  $r_\tau = 0$  then
5:      $\mathcal{F} \leftarrow \mathcal{F} \cup \{(\tau, \xi_\tau)\}$   $\triangleright$  failure case
6:   end if
7: end for
8: for each  $(\tau, \xi) \in \mathcal{F}$  do
9:   for each error category  $e \in \mathcal{E}$  do
10:     $o_{\tau,e} \leftarrow \text{ANALYZE}_e(\xi)$   $\triangleright$  independent
    error-analysis agent for  $e$ 
11:   end for
12:    $O_\tau \leftarrow \text{CONCAT}(\{o_{\tau,e}\}_{e \in \mathcal{E}})$ 
13:    $\hat{\mathcal{E}}_\tau \leftarrow \text{ORCHESTRATE}(O_\tau, \xi)$   $\triangleright$  main
   failure error(s)
14:    $\mathcal{A}_\tau^* \leftarrow \text{MITIGATE}(\hat{\mathcal{E}}_\tau, \mathcal{A})$   $\triangleright$  optimal agent
   subset
15: end for
16: return  $\{\mathcal{A}_\tau^*\}_{\tau \in \mathcal{T}}$ 

```

Qwen3-32B, and **Qwen2.5-72B-Instruct**. We exclude reasoning- or thinking-augmented variants from our study, as they fall outside the scope of this work. Our analysis indicates that such models consume a substantial portion of their token budget in the internal reasoning process, which introduces token limit constraints when deployed within agentic frameworks. As a result, these models tend to exhibit inferior performance in tool-calling settings.

Methods. To assess the effectiveness of FAMA, we compare it against several established baselines, including **Function Calling (FC)**, **ReAct**, and the **Input-Reformulation Multi-Agent framework (IRMA)**. The IRMA study also investigates an enhanced variant incorporating a Self-Reflection mechanism, and reports improved performance over the standard IRMA setup. Since this extension is orthogonal to our focus, we do not include it in our experiments and leave its integration with FAMA for future work.

Evaluation. In this work, we focus on dynamic environments in which a user-agent simulator interacts with a tool-calling agent. Accordingly, we evaluate all methods on three benchmarks. First,

Method	τ -Airline					τ -Retail				
	Pass^1	Pass^2	Pass^3	Pass^4	Pass^5	Pass^1	Pass^2	Pass^3	Pass^4	Pass^5
<i>Qwen3-4B-Instruct</i>										
ReAct	32.00%	28.00%	26.80%	26.40%	26.00%	17.22%	12.35%	10.61%	9.57%	8.70%
FC	27.60%	19.90%	17.00%	15.20%	14.00%	24.90%	16.70%	13.00%	10.60%	9.00%
IRMA	30.00%	19.60%	15.40%	13.20%	12.00%	28.86%	17.47%	13.13%	10.95%	9.56%
FAMA (our)	37.60%	32.00%	28.30%	26.70%	26.00%	34.60%	24.10%	19.30%	16.30%	13.90%
<i>Qwen3-14B</i>										
ReAct	18.40%	12.00%	9.40%	8.40%	8.00%	25.20%	17.80%	14.70%	13.20%	12.10%
FC	14.00%	9.60%	8.40%	8.00%	8.00%	32.60%	22.00%	18.00%	15.00%	13.00%
IRMA	36.40%	26.40%	22.00%	19.60%	18.00%	28.50%	15.60%	10.80%	8.30%	6.90%
FAMA (our)	36.80%	23.79%	18.20%	16.60%	16.00%	37.90%	25.70%	19.70%	16.30%	14.70%
<i>Qwen3-32B</i>										
ReAct	30.40%	20.00%	16.20%	14.80%	14.00%	38.00%	25.00%	18.80%	15.30%	10.00%
FC	17.60%	13.60%	12.00%	10.80%	10.00%	35.00%	26.60%	15.00%	13.10%	11.00%
IRMA	24.40%	15.90%	12.40%	10.00%	8.00%	26.90%	15.20%	10.90%	8.50%	6.90%
FAMA (our)	26.80%	20.00%	18.40%	18.00%	18.00%	40.50%	26.90%	19.90%	15.30%	12.20%
<i>Qwen2.5-72B-Instruct</i>										
ReAct	24.40%	18.79%	15.60%	12.80%	10.00%	43.47%	32.26%	26.69%	23.13%	20.86%
FC	15.20%	7.60%	4.00%	2.40%	2.00%	19.65%	10.34%	6.78%	5.21%	4.34%
IRMA	26.40%	16.70%	13.60%	11.60%	10.00%	38.78%	27.73%	22.95%	20.34%	19.13%
FAMA (our)	29.20%	21.20%	18.80%	18.00%	18.00%	44.17%	34.26%	30.26%	28.17%	26.95%

Table 1: Comparison of different open-source agents on τ -bench across five runs. Note that Qwen2.5-72B-Instruct is used as the model backbone of the sub-agents in the IRMA and FAMA agentic frameworks.

we use the τ -bench, with evaluations conducted in the Airline and Retail domains. Second, we consider τ -trait, where performance is assessed in the Telehealth and Telecom domains. Finally, we evaluate the methods on the single-agent setting of ACEBench, which covers tasks related to food delivery services and telecommunication operations. For detailed information about the benchmarks and evaluation metrics, refer to Appendix B.

We divide our experiments into two stages. First, we identify the best-performing open-source model to serve as the user agent. Next, we evaluate different methods across multiple open-source tool-calling models while fixing the user agent to the selected best-performing model. We additionally note that **GPT-4o** (Achiam et al., 2023) and **GPT-4.1-mini** are used as the judgment agents within the FAMA framework to demonstrate the consistency of the judgments across different models (see Appendix A).

5.2 Experimental Results

Open-source agents can be trusted as user simulator agents. In this experiment, we use the Qwen3-4B-Thinking model as the tool-calling agent. We select this model due to its relatively small size and its ability to achieve performance comparable to larger models. For the

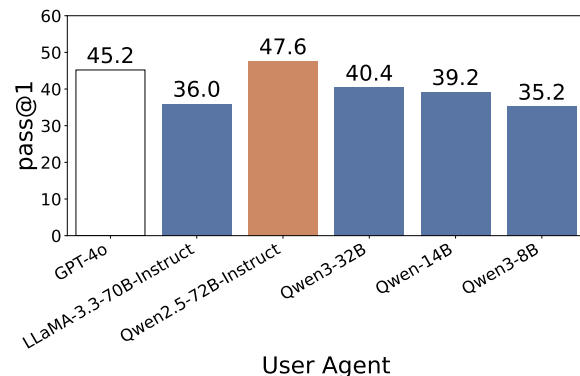


Figure 3: Comparing τ -bench performance across multiple LLMs being tested as user agents.

user simulator, we evaluate five state-of-the-art open-source models, namely LLaMA-3.1-70B-Instruct (Grattafiori et al., 2024), Qwen2.5-72B-Instruct, Qwen3-8B, Qwen3-14B, and Qwen3-32B (Yang et al., 2025). All experiments are conducted on the Airline domain of τ -bench, and the results are compared against a setting in which GPT-4o is used as the user agent. The results in Figure 3 indicate that, among open-source models, Qwen2.5-72B-Instruct is the closest to a frontier model such as GPT-4o, as its pass@1 performance averaged over five runs surpasses that of GPT-4o.

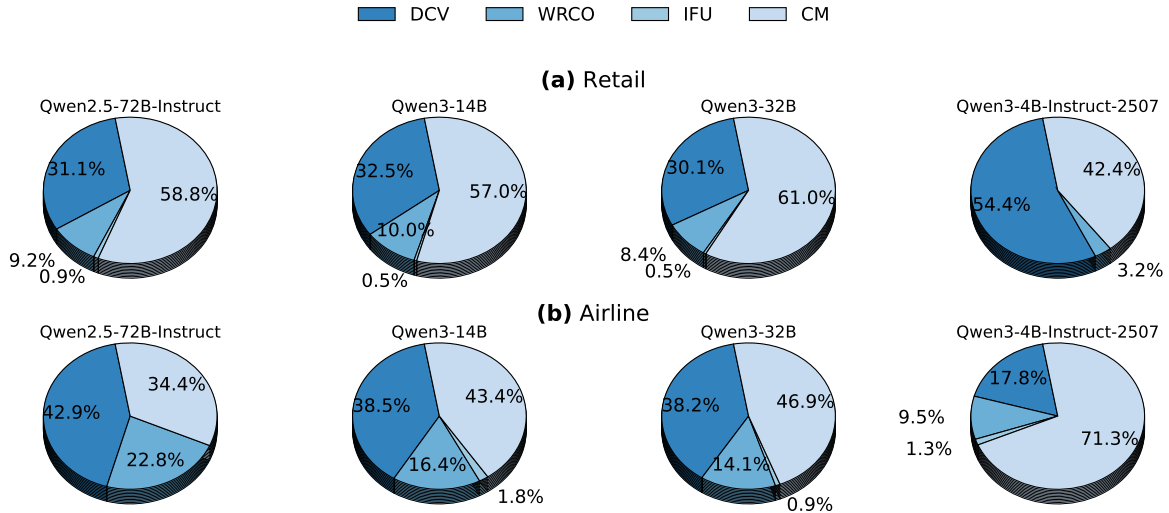


Figure 4: Statistical comparison of error categories across open-source LLMs in τ -retail, as identified by the Orchestrator Agent.

Using all agents leads to inconsistent performance across models. Consistent with prior work, we evaluate open-source models on τ -bench in the IRMA setting. Table 1 shows that IRMA outperforms other baselines in a limited number of cases, indicating that while multi-agent frameworks can be beneficial, indiscriminately using all agents does not consistently improve performance across models and can, in some cases, degrade the base agent’s performance.

FAMA consistently outperforms baseline methods across different benchmarks. Tables 1–4 show that the effectiveness of an agentic framework is highly sensitive to both agent selection and domain, and that naively including all agents can degrade performance. By selectively enabling agents based on dominant failure modes, FAMA avoids unnecessary context injection and focuses mitigation on the most impactful errors. As shown in Table 1, FAMA outperforms ReAct, FC, and IRMA by 4.63%, 11.57%, and 5.27% in the Airline domain and by 5.30%, 8.96%, and 6.15% in the Retail domain, averaged over five runs.

FAMA is a generalizable method. Generalizability is a key limitation of static multi-agent frameworks, which are typically tailored to frontier-model failures in specific domains and require extensive manual redesign to adapt to changes. In contrast, FAMA is a dynamic framework that overcomes this limitation. Experiments across multiple domains and benchmarks, including ACEBench

and τ -trait, show that FAMA consistently outperforms other methods by adapting to agent failure behavior (See Tables 3 and 4). For detailed results on the τ -trait and ACEBench benchmarks, refer to Appendix D and C. The modular components in FAMA enable the easy addition of specialized agents to address specific error patterns.

To further contextualize our results, we also evaluate reasoning-enabled (“thinking”) variants of Qwen3 models. As shown in Figure 6, while such variants can improve certain baseline methods, they incur higher token usage (see Table 2) and frequently lead to context window overflows in multi-turn settings, where accumulated reasoning traces consume a significant portion of the available token budget. We quantify this effect by counting instances (refer to Table 8) that exceed the maximum token limit across runs and treat them as failures, as mitigating them would require truncating critical context (e.g., domain constraints or prior tool outputs), potentially leading to incorrect decisions. In contrast, FAMA achieves stronger and more stable performance across k without relying on expensive internal reasoning steps.

5.3 Analysis and Ablation Study

To analyze error patterns, we run each task within a domain five times. The results in Figures 4, 12, and 14 show that all evaluated open-source models exhibit significant difficulty with domain constraint violations and contextual misinterpretations. This indicates that, even when solving a task, select-

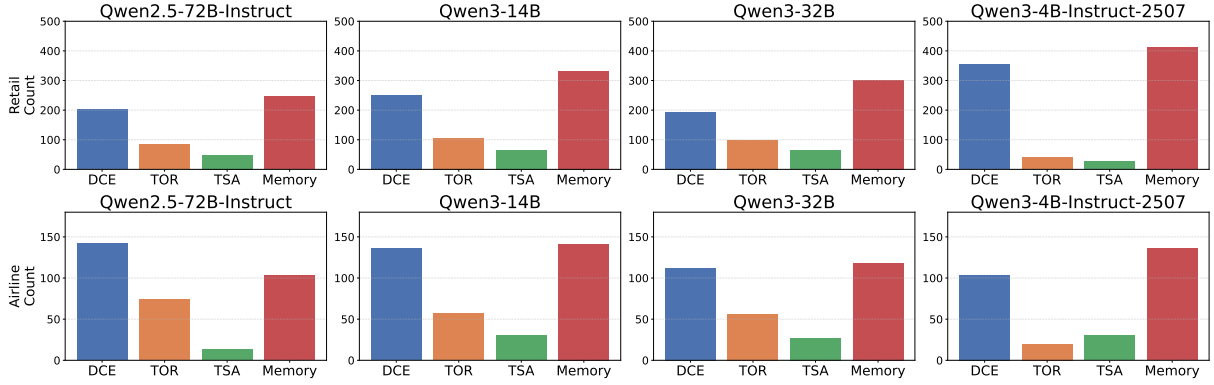


Figure 5: Statistics of recommendations produced by the Mitigation Agent across open source agents for τ -bench.

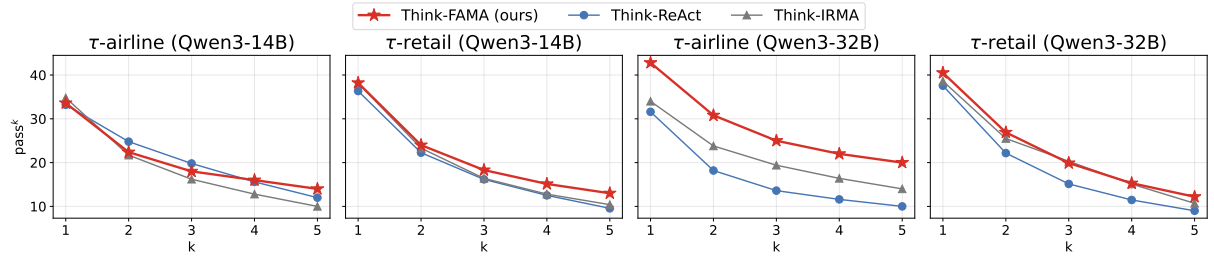


Figure 6: Performance comparison of Qwen3-14B and Qwen3-32B (thinking variants) across methods on τ -bench.

ing actions that strictly adhere to domain-specific rules remains challenging. This issue becomes more pronounced as conversations grow longer, since domain constraints provided in the system prompt tend to be forgotten over time, highlighting memory-related limitations in open-source models. The results in Figure 5 show that the mitigation agent strongly recommends the Memory module and the DCE agent, confirming that memory is a critical bottleneck for open-source agents.

Motivated by these findings, we conduct an ablation study on memory size to identify effective configurations across domains. As shown in Appendix F, the optimal memory size is domain-dependent rather than model-dependent. Specifically, Figure 10 shows that the Retail domain, which involves longer and more complex user-agent interactions, benefits from a larger memory size, with $k = 6$ achieving the best performance, whereas the Airline domain attains optimal results with a smaller memory size of $k = 2$. We also compare these settings against a no-memory baseline. Overall, the results demonstrate that an agentic framework with an appropriately sized memory not only outperforms IRMA, which uses all agents, but also surpasses other baseline methods. Additional ablation studies on agent combinations that differ from those recommended by the mitigation agent,

as well as evaluations with various frontier models in the FAMA framework, have been provided in Appendix A. A comparison with an additional baseline, Self-Reflection (based on Madaan et al. (2023)), is provided in Appendix G.

We also experiment and analyze efficiency in terms of latency and token overhead (see Figure 7) across various baselines and our method. Methods that introduce additional reasoning or helper agents incur higher runtime and token costs, which can worsen context-window size constraints in multi-turn settings. In particular, IRMA exhibits substantial overhead (50 and 58%), and task completion latency (149.8 and 111.6 seconds averaged) for Qwen3-32B on retail and airline tasks. While FAMA also incurs additional cost compared to ReAct baselines, its more efficient design ($\sim 30\%$ overhead) results in lower latency than IRMA and fewer overflow-induced errors. Although absolute latency depends on system and deployment configurations, the relative differences across methods are consistent, with IRMA incurring the highest cost. Overall, this suggests that the added cost of agentic scaffolding offsets its benefits by straining the available token budget and degrading overall reliability.

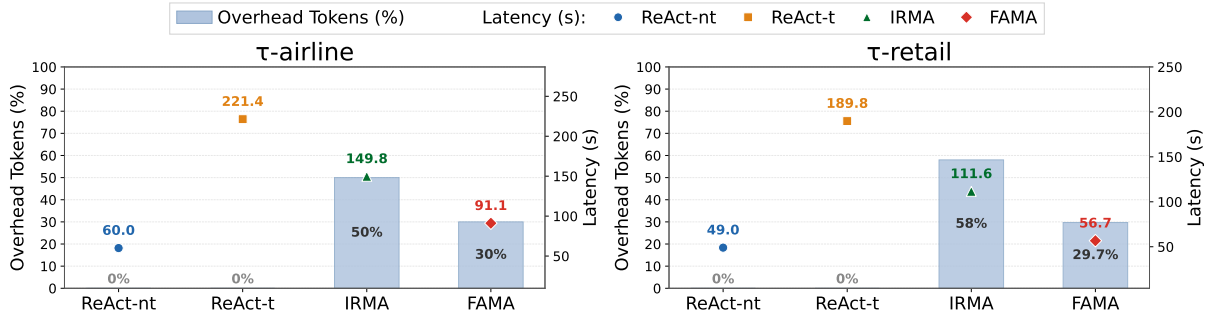


Figure 7: Token overhead (%) and latency (seconds) across different methods with Qwen3-32B. See Table 2 for detailed statistics. ‘ReAct-nt’ and ‘ReAct-t’ represent non-thinking and thinking models used for ReAct method.

6 Conclusion

In this work, we present the Failure-Aware Meta-Agentic (FAMA) framework, a dynamic multi-agent approach designed to overcome the limitations of open-source language agents in interactive environments such as tool calling. FAMA operates in two stages: first, it analyzes agent failures to identify dominant error patterns; second, it selects a minimal, task-relevant subset of agents via an orchestrator framework to mitigate these errors. We evaluate FAMA across three interactive tool-calling benchmarks and four open-source LLMs. Results and analyses demonstrate the potential of FAMA as a generalizable, failure-aware orchestration framework for constructing optimized prior contexts as a key design principle for improving and building dependable, resource-efficient conversational agents.

Limitations

While the proposed Failure-Aware Meta-Agentic (FAMA) framework achieves consistent improvements across multiple benchmarks, several limitations remain.

First, FAMA operates over a predefined pool of specialized agents inherited from the underlying agentic framework (§4.1). This design reflects a common setting in existing user-facing, modular tool-use systems, where a fixed set of components is orchestrated rather than dynamically generated. As a result, FAMA’s effectiveness is bounded by the coverage of this agent pool: failure modes that are not well captured by existing agents may not be adequately mitigated. Importantly, this assumption does not imply rigidity. FAMA is designed to be extensible. New specialized agents targeting emerging failure patterns can be incorporated without modifying the orchestration mechanism. However, the current framework does not address the

automatic discovery or synthesis of such agents, which remains an important direction for future work.

Furthermore, the benchmarks considered in this study primarily focus on structured conversational environments. While these settings provide a controlled testbed for analyzing failure-aware orchestration, they do not capture the full spectrum of interactive agent deployments. Extending FAMA to more diverse settings, including embodied, multi-modal, or open-ended environments with less well-defined failure taxonomies, is an important avenue for future investigation.

Ethics Statement

We have utilized AI assistants, specifically Grammarly and ChatGPT, to correct grammatical errors and rephrase sentences.

Acknowledgement

We thank the anonymous reviewers for their constructive suggestions. We extend our gratitude to the Research Computing (RC), and Enterprise Technology at ASU for providing computing resources, and access to the GPT API version for experiments. This work was in part supported by a gift award from Cisco Research. This work makes use of the τ -bench, τ -trait, and ACEBench dataset, which are distributed under the MIT License.

References

- Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, and 1 others. 2024. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*.
- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman,

- Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- Emre Can Acikgoz, Jinh Oh, Jie Hao, Joo Hyuk Jeon, Heng Ji, Dilek Hakkani-Tür, Gokhan Tur, Xiang Li, Chengyuan Ma, and Xing Fan. 2025. Speakrl: Synergizing reasoning, speaking, and acting in language models with reinforcement learning. *arXiv preprint arXiv:2512.13159*.
- Mariana Almeida. 2025. [Building realistic multi-turn tests for ai agents](#). Accessed: 2026-01-02.
- Anthropic. 2025. [Introducing claude opus 4.5](#). Accessed: 2026-01-02.
- Neveen Awad, Mahmood Serry, and Joe Vasquez. 2025. [How agentic ai is transforming enterprise platforms](#). Accessed: 2026-01-03.
- Adi Banerjee, Anirudh Nair, and Tarik Borogovac. 2025. Where did it all go wrong? a hierarchical look into multi-agent error attribution. *arXiv preprint arXiv:2510.04886*.
- Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. 2025. [\$\tau\$ 2-bench: Evaluating conversational agents in a dual-control environment](#). *ArXiv*, abs/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*.
- Mert Cemri, Melissa Z Pan, Shuyi Yang, Lakshya A Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, and 1 others. 2025. Why do multi-agent llm systems fail? *arXiv preprint arXiv:2503.13657*.
- Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. 2023. Fireact: Toward language agent fine-tuning. *arXiv preprint arXiv:2310.05915*.
- Chen Chen, Xinlong Hao, Weiwen Liu, Xu Huang, Xingshan Zeng, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Yuefeng Huang, and 1 others. 2025a. Acebench: Who wins the match point in tool usage? *arXiv preprint arXiv:2501.12851*.
- Zhaorun Chen, Zhuokai Zhao, Kai Zhang, Bo Liu, Qi Qi, Yifan Wu, Tarun Kalluri, Sara Cao, Yuanhao Xiong, Haibo Tong, and 1 others. 2025b. Scaling agent learning via experience synthesis. *arXiv preprint arXiv:2511.03773*.
- Sadia Sultana Chowh, Riasad Alvi, Subhey Sadi Rahman, Md Abdur Rahman, Mohaimenul Azam Khan Raiaan, Md Rafiqul Islam, Mukhtar Hussain, and Sami Azam. 2025. From language to action: A review of large language models as autonomous agents and tool users. *arXiv preprint arXiv:2508.17281*.
- Yihong Dong, Xue Jiang, Jiaru Qian, Tian Wang, Kechi Zhang, Zhi Jin, and Ge Li. 2025. A survey on code generation with llm-based agents. *arXiv preprint arXiv:2508.00083*.
- Nicholas Farn and Richard Shin. 2023. Tooltalk: Evaluating tool-usage in a conversational setting. *arXiv preprint arXiv:2311.10775*.
- Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. 2024. Autoguide: Automated generation and selection of context-aware guidelines for large language model agents. *Advances in Neural Information Processing Systems*, 37:119919–119948.
- Yu Ge, Linna Xie, Zhong Li, Yu Pei, and Tian Zhang. 2025. Who is introducing the failure? automatically attributing failures of multi-agent systems via spectrum analysis. *arXiv preprint arXiv:2509.13782*.
- Alexander Golubev, Maria Trofimova, Sergei Polezhaev, Ibragim Badertdinov, Maksim Nekrashevich, Anton Shevtsov, Simon Karasik, Sergey Abramov, Andrei Andriushchenko, Filipp Fisin, and 1 others. 2025. Training long-context, multi-turn software engineering agents with reinforcement learning. *arXiv preprint arXiv:2508.03501*.
- Google DeepMind. 2025. [Gemini 3 pro](#). Accessed: 2026-01-02.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Divij Handa, David Blincoe, Orson Adams, and Yinlin Fu. 2025a. Optagent: Optimizing query rewriting for e-commerce via multi-agent simulation. *arXiv preprint arXiv:2510.03771*.
- Divij Handa, Mihir Parmar, Aswin RRV, Md Nayem Uddin, Hamid Palangi, and Chitta Baral. 2025b. Guidedsampling: Steering llms towards diverse candidate solutions at inference-time. *arXiv preprint arXiv:2510.03777*.
- Muyu He, Anand Kumar, Tsach Mackey, Meghana Rajeev, James Zou, and Nazneen Rajani. 2025. [Impatient users confuse ai agents: High-fidelity simulations of human traits for testing agents](#). *arXiv preprint arXiv:2510.04491*.
- Yuchen Huang, Sijia Li, Minghao Liu, Wei Liu, Shijue Huang, Zhiyuan Fan, Hou Pong Chan, and Yi R Fung. 2025. Environment scaling for interactive agentic experience collection: A survey. *arXiv preprint arXiv:2511.09586*.
- Polaris Jhandi, Owais Kazi, Shreyas Subramanian, and Neel Sendas. 2025. Small language models for efficient agentic tool calling: Outperforming large models with targeted fine-tuning. *arXiv preprint arXiv:2512.15943*.

- Dongfu Jiang, Yi Lu, Zhuofeng Li, Zhiheng Lyu, Ping Nie, Haozhe Wang, Alex Su, Hui Chen, Kai Zou, Chao Du, and 1 others. 2025. Verltool: Towards holistic agentic reinforcement learning with tool use. *arXiv preprint arXiv:2509.01055*.
- Kiran Kate, Tejaswini Pedapati, Kinjal Basu, Yara Rizek, Vijil Chenthamarakshan, Subhajit Chaudhury, Mayank Agarwal, and Ibrahim Abdelaziz. 2025. Longfunceval: Measuring the effectiveness of long context models for function calling. *arXiv preprint arXiv:2505.10570*.
- Kai-Kristian Kemell, Matti Saarikallio, Anh Nguyen-Duc, and Pekka Abrahamsson. 2025. Still just personal assistants? – a multiple case study of generative ai adoption in software organizations. *Information and Software Technology*, 186:107805.
- Shirley Kokane, Ming Zhu, Tulika Awalgaonkar, Jianguo Zhang, Thai Hoang, Akshara Prabhakar, Zuxin Liu, Tian Lan, Liangwei Yang, Juntao Tan, and 1 others. 2024. Spectool: A benchmark for characterizing errors in tool-use llms. *arXiv preprint arXiv:2411.13547*.
- Shrinidhi Kumbhar, Venkatesh Mishra, Kevin Coutinho, Divij Handa, Ashif Iqbal, and Chitta Baral. 2025. Hypothesis generation for materials discovery and design using goal-driven and constraint-guided LLM agents. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 7539–7570, Albuquerque, New Mexico. Association for Computational Linguistics.
- Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. API-bank: A comprehensive benchmark for tool-augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3102–3116, Singapore. Association for Computational Linguistics.
- Guannan Liang and Qianqian Tong. 2025. Llm-powered ai agent systems and their applications in industry. In *2025 IEEE World AI IoT Congress (AlloT)*, pages 0463–0471.
- Ilija Lichkovski, Alexander Müller, Mariam Ibrahim, and Tiwai Mhundwa. 2025. Eu-agent-bench: Measuring illegal behavior of llm agents under eu law. *arXiv preprint arXiv:2510.21524*.
- Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, and 1 others. 2025a. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*.
- Shuo Liu, Kaining Ying, Hao Zhang, Yue Yang, Yuqi Lin, Tianle Zhang, Chuanhao Li, Yu Qiao, Ping Luo, Wenqi Shao, and 1 others. 2024. Convbench: A multi-turn conversation evaluation benchmark with hierarchical capability for large vision-language models. *arXiv preprint arXiv:2403.20194*.
- Zihan Liu, Yang Chen, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2025b. Acemath: Advancing frontier math reasoning with post-training and reward modeling. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 3993–4015.
- Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, and 1 others. 2025a. Toolsandbox: A stateful, conversational, interactive evaluation benchmark for llm tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1160–1183.
- Ruofan Lu, Yichen Li, and Yintong Huo. 2025b. Exploring autonomous agents: A closer look at why they fail when completing tasks. *arXiv preprint arXiv:2508.13143*.
- Zhiyuan Ma, Jiayu Liu, Xianzhen Luo, Zhenya Huang, Qingfu Zhu, and Wanxiang Che. 2025. Advancing tool-augmented large language models via meta-verification and reflection learning. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2, KDD '25*, page 2078–2089, New York, NY, USA. Association for Computing Machinery.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.
- Venkatesh Mishra, Amir Saeidi, Satyam Raj, Mutsumi Nakamura, Jayanth Srinivasa, Gaowen Liu, Ali Payani, and Chitta Baral. 2025. How can input reformulation improve tool usage accuracy in a complex dynamic environment? a study on τ -bench. *ArXiv*, abs/2508.20931.
- Hadi Nekoei, Aman Jaiswal, Patrice Bechard, Oleh Shliazhko, Orlando Marquez Ayala, Mathieu Raymond, Massimo Caccia, Alexandre Drouin, Sarath Chandar, and Alexandre Lacoste. 2025. Just-in-time episodic feedback hinter: Leveraging offline knowledge to improve llm agents adaptation. *arXiv preprint arXiv:2510.04373*.
- OpenAI. 2025. Introducing gpt-5.2. Accessed: 2026-01-02.
- Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. 2024. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37:126544–126565.
- Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Jiadai Sun, Xinyue Yang, Yu Yang, Shuntian Yao, Wei Xu, and 1 others. 2025. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. In *The Thirteenth International Conference on Learning Representations*.

- 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*.
- Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. 2025. Tool learning with large language models: A survey. *Frontiers of Computer Science*, 19(8):198343.
- Mizanur Rahman, Amran Bhuiyan, Mohammed Saidul Islam, Md Tahmid Rahman Laskar, Ridwan Mahbub, Ahmed Masry, Shafiq Joty, and Enamul Hoque. 2025. Llm-based data science agents: A survey of capabilities, challenges, and future directions. *arXiv preprint arXiv:2510.04023*.
- Paulius Rauba, Nabeel Seedat, Max Ruiz Luyten, and Mihaela van der Schaar. 2024. [Context-aware testing: A new paradigm for model testing with large language models](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 112505–112553. Curran Associates, Inc.
- Aswin Rrv, Jacob Dineen, Divij Handa, Md Nayem Uddin, Mihir Parmar, Chitta Baral, and Ben Zhou. 2025. [ThinkTuning: Instilling cognitive reflections without distillation](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 31236–31250, Suzhou, China. Association for Computational Linguistics.
- Amir Saeidi, Chitta Baral, Ahmed Hassan Awadallah, and Harkirat Behl. 2026. Vulcan: Where agents learn by living in simulated tool environments. In *ICLR 2026 Workshop on Data-FM*. Last modified: April 1, 2026.
- Sarath Shekkizhar, Romain Cosentino, Adam Earle, and Silvio Savarese. 2025. Echoing: Identity failures when llm agents talk to each other. *arXiv preprint arXiv:2511.09710*.
- Weizhou Shen, Chenliang Li, Hongzhan Chen, Ming Yan, Xiaojun Quan, Hehong Chen, Ji Zhang, and Fei Huang. 2024. [Small LLMs are weak tool learners: A multi-LLM agent](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16658–16680, Miami, Florida, USA. Association for Computational Linguistics.
- 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.
- Akshit Sinha, Arvinth Arun, Shashwat Goel, Steffen Staab, and Jonas Geiping. 2025. The illusion of diminishing returns: Measuring long horizon execution in llms. In *First Workshop on Multi-Turn Interactions in Large Language Models*.
- Peter Stone and Manuela Veloso. 2000. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383.
- Hongru Wang, Wenyu Huang, Yufei Wang, Yuanhao Xi, Jianqiao Lu, Huan Zhang, Nan Hu, Zeming Liu, Jeff Z. Pan, and Kam-Fai Wong. 2025a. [Rethinking stateful tool use in multi-turn dialogues: Benchmarks and challenges](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 5433–5453, Vienna, Austria. Association for Computational Linguistics.
- Hongru Wang, Wenyu Huang, Yufei Wang, Yuanhao Xi, Jianqiao Lu, Huan Zhang, Nan Hu, Zeming Liu, Jeff Z Pan, and Kam-Fai Wong. 2025b. Rethinking stateful tool use in multi-turn dialogues: Benchmarks and challenges. *arXiv preprint arXiv:2505.13328*.
- Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. 2023. Mint: Evaluating llms in multi-turn interaction with tools and language feedback. *arXiv preprint arXiv:2309.10691*.
- Yutong Wang, Pengliang Ji, Kaixin Li, Baolong Bi, Tao Feng, and Guillaume Sartoretti. 2025c. Beyond policy optimization: A data curation flywheel for sparse-reward long-horizon planning. *arXiv preprint arXiv:2508.03018*.
- Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. 2025d. Agent workflow memory. In *Forty-second International Conference on Machine Learning*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.
- Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. 2024. [\$\tau\$ -bench: A benchmark for tool-agent-user interaction in real-world domains](#). *Preprint*, arXiv:2406.12045.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.
- Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2024. [AgentTuning: Enabling generalized agent abilities for LLMs](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3053–3077, Bangkok, Thailand. Association for Computational Linguistics.

- Yirong Zeng, Xiao Ding, Yutai Hou, Yuxian Wang, Li Du, Juyi Dai, Qiuyang Ding, Duyu Tang, Dandan Tu, Weiwen Liu, Bing Qin, and Ting Liu. 2025. [Tool zero: Training tool-augmented LLMs via pure RL from scratch](#). In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 9135–9147, Suzhou, China. Association for Computational Linguistics.
- Chen Zhang, Xinyi Dai, Yaxiong Wu, Qu Yang, Yasheng Wang, Ruiming Tang, and Yong Liu. 2025a. A survey on multi-turn interaction capabilities of large language models. *arXiv preprint arXiv:2501.09959*.
- Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, Yijiang Li, and 1 others. 2025b. The landscape of agentic reinforcement learning for llms: A survey. *arXiv preprint arXiv:2509.02547*.
- Guibin Zhang, Junhao Wang, Junjie Chen, Wangchunshu Zhou, Kun Wang, and Shuicheng Yan. 2025c. Agentracer: Who is inducing failure in the llm agentic systems? *arXiv preprint arXiv:2509.03312*.
- Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, and 1 others. 2025d. Agent learning via early experience. *arXiv preprint arXiv:2510.08558*.
- Wentao Zhang, Ce Cui, Yilei Zhao, Yang Liu, and Bo An. 2025e. Agentorchestra: A hierarchical multi-agent framework for general-purpose task solving. *arXiv preprint arXiv:2506.12508*.
- Bingxi Zhao, Lin Geng Foo, Ping Hu, Christian Theobalt, Hossein Rahmani, and Jun Liu. 2025a. Llm-based agentic reasoning frameworks: A survey from methods to scenarios. *arXiv preprint arXiv:2508.17692*.
- Weikang Zhao, Xili Wang, Chengdi Ma, Lingbin Kong, Zhaohua Yang, Mingxiang Tuo, Xiaowei Shi, Yitao Zhai, and Xunliang Cai. 2025b. [Mua-rl: Multi-turn user-interacting agent reinforcement learning for agentic tool use](#). *ArXiv*, abs/2508.18669.
- Lianhao Zhou, Hongyi Ling, Cong Fu, Yepeng Huang, Michael Sun, Wendi Yu, Xiaoxuan Wang, Xiner Li, Xingyu Su, Junkai Zhang, and 1 others. 2025. Autonomous agents for scientific discovery: Orchestrating scientists, language, code, and physics. *arXiv preprint arXiv:2510.09901*.
- Kunlun Zhu, Zijia Liu, Bingxuan Li, Muxin Tian, Yingxuan Yang, Jiaxun Zhang, Pengrui Han, Qipeng Xie, Fuyang Cui, Weijia Zhang, and 1 others. 2025. Where llm agents fail and how they can learn from failures. *arXiv preprint arXiv:2509.25370*.
- K.J Åström. 1965. [Optimal control of markov processes with incomplete state information](#). *Journal of Mathematical Analysis and Applications*, 10(1):174–205.

A More Ablation Studies

We conduct additional ablation studies on agent combinations that are not recommended by the mitigation agent, as well as on different judgment models, to assess the quality of the resulting decisions and recommendations.

Alternative agent combinations exhibit inferior performance. In this section, we examine the performance of alternative agent combinations that were not recommended by the mitigation agent, in order to further demonstrate the effectiveness of the FAMA method. Specifically, we select Qwen3-14B as the tool-calling agent and conduct Tau-Bench experiments using the following configurations:

Exp 1: Memory module, Domain Constraints Extractor Agent, and Tool Output Reformulation Agent.

Exp 2: Memory module and Tool Output Reformulation Agent.

Exp 3: Memory module, Tool Output Reformulation Agent, and Tool Suggestion Agent.

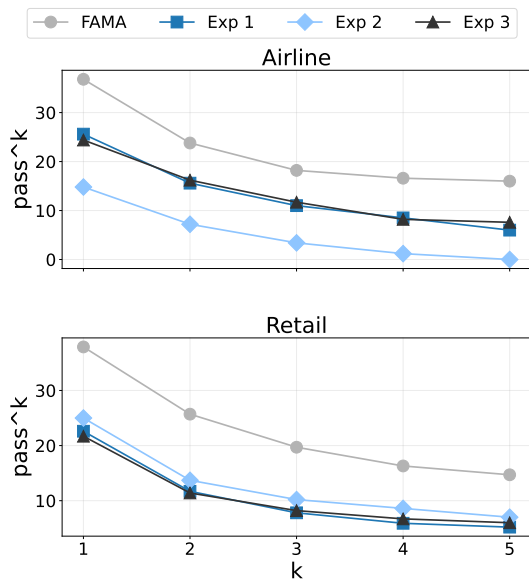


Figure 8: Comparison of tool calling agents that rely on different agentic combinations.

The results in Figure 8 show that the best performance is achieved by the agent combination recommended by the mitigation agent. We repeat this analysis across other models and observe the same trend, further demonstrating the generalizability of FAMA. We also find that, in some cases, using only a subset of the agents recommended by the mitigation agent can yield competitive or even

improved performance. Overall, these results highlight the effectiveness of the FAMA framework in selecting appropriate agent combinations.

Different judgment models produce the same decisions. For failure analysis and agent selection within the FAMA framework, we initially use GPT-4o. To evaluate the robustness of the decision process, we repeat the same analysis and selection procedure on baseline results using GPT-4.1-mini as the judgment model. The results in Figure 9 show that, similar to GPT-4o, the alternative judgment model identifies Contextual Misinterpretation (CM) and Domain Constraint Violation (DCV) as the primary failure modes of open-source models. Accordingly, the mitigation agent consistently recommends the Memory module and the Domain Constraints Extractor agent, as shown in Figure 11.

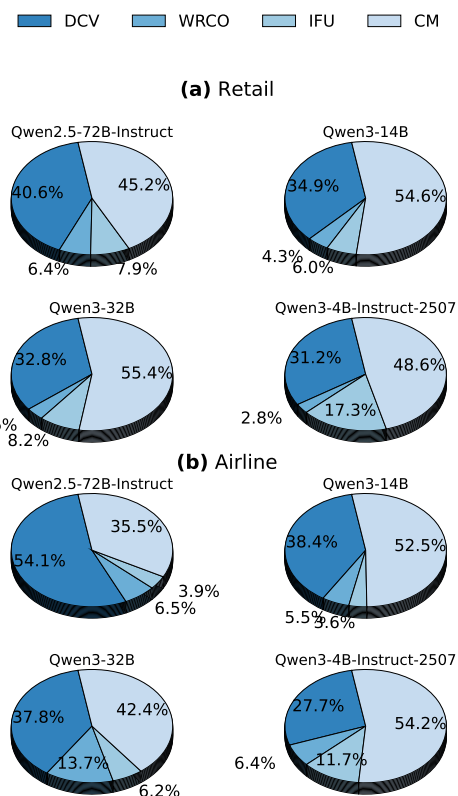


Figure 9: Statistical comparison of error categories across open-source LLMs in τ -bench, as identified by the Orchestrator Agent using GPT-4.1-mini.

These findings indicate that different frontier models exhibit consistent behavior in failure analysis and agent selection within the FAMA framework.

τ -bench-airline								
Method	Min Tokens	Max Tokens	Median Tokens	Avg Tokens	Assistant Tokens	Overhead Tokens	Overhead (%)	Latency (seconds)
React-non-thinking	96	4591	971.5	1148.9	1148.9	0	0	59.99
React-thinking	269	32175	4957.5	5729.6	5729.6	0	0	221.43
IRMA	346	14324	2725	3511.1	1694.4	1816.7	50	149.78
FAMA	358	8111	2137	2617.4	1822.1	795.3	30	91.12

τ -bench-retail								
Method	Min Tokens	Max Tokens	Median Tokens	Avg Tokens	Assistant Tokens	Overhead Tokens	Overhead (%)	Latency (seconds)
React-non-thinking	59	3727	1108	1131.4	1131.4	0	0	49.05
React-thinking	246	32768	4331	4623.4	4623.4	0	0	189.79
IRMA	380	32768	3421	3980.4	1671.72	2308.7	58	111.62
FAMA	161	8040	2167	2440	1714.6	725.4	29.7	56.73

Table 2: Token and latency cost analysis of various methods for τ -Airline and τ -Retail. The tokens reported in Columns 2–5 refer to the sum of (i) tokens generated by the main tool-use assistant (Column 6) and (ii) additional overhead tokens introduced by helper agents (Column 7). ‘Assistant tokens’ denote the average tokens produced solely by the primary tool-calling agent, while ‘overhead tokens’ correspond to the average tokens generated by helper agents in the IRMA and FAMA settings. ‘ReAct-non-thinking’ and ‘ReAct-thinking’ represent non-thinking and thinking models used for ReAct method.

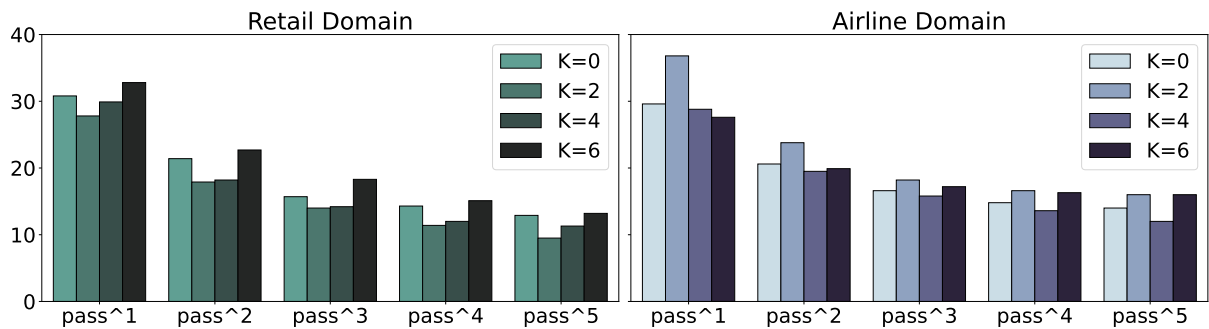


Figure 10: Comparison of agent performance in τ -bench across different values of k , where k denotes the number of most recent user queries retained.

B Detail of Benchmarks

Each task in τ -bench, τ -trait and ACEBench is formulated as a Partially Observable Markov Decision Process (POMDP) (Åström, 1965), in which an assistant agent interacts with a user entity/agent through a sequence of natural language conversations and tool invocations. At each step, the agent must decide which function calls to issue based on the current conversational context and partial observations derived from the external environment. Executed function calls produce structured outputs that are incorporated into the evolving dialogue and inform subsequent decisions. This interaction proceeds over multiple turns until task termination, at which point the agent is evaluated using the appropriate benchmark-prescribed metric centered around successful conversation/task completion.

τ -bench τ -bench evaluates tool using conversational agents in two real-world-style customer-service domains—retail and airline, where an agent must interact with a simulated user, follow domain policy rules, and use API tools to update/query a realistic backend state. In retail, the benchmark includes a database with 500 users, 50 products, and 1,000 orders, exposes 15 tools total (7 write + 8 non-write), and provides 115 tasks spanning actions like cancellations, returns/exchanges, address changes, and information requests; in airline, it includes 500 users, 300 flights, and 2,000 reservations, has 13 tools (6 write + 7 non-write), and contains 50 tasks around booking/modifying/canceling reservations and refund-related workflows.

τ -trait τ -trait is a robustness-focused extension of τ -bench that keeps the multi-turn tool-use setting but adds persona-aware user simulation and

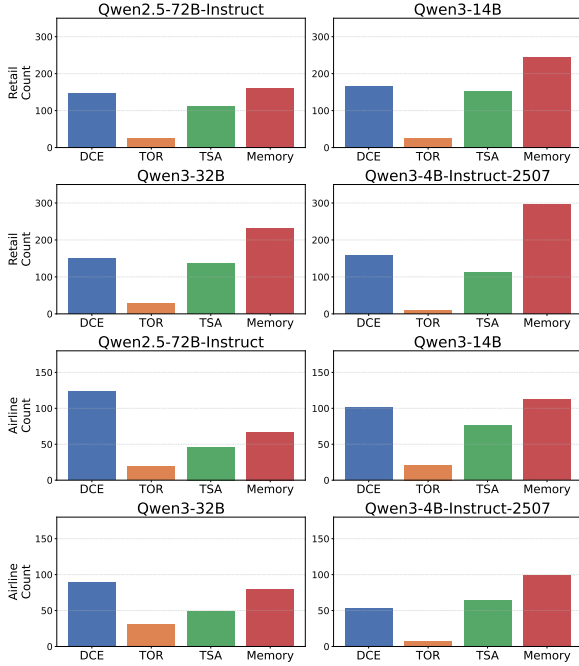


Figure 11: Statistics of recommendations produced by the Mitigation Agent using GPT-4.1-mini across different open source tool-use agents.

expands domain coverage beyond retail/airline to include telecom and telehealth, targeting robustness, personalization, and fairness under realistic behavior shifts. It contains 218 total tasks split across 4 distinct domains, like retail (120), airline (60), telecom (18), and telehealth (20).

For τ -bench and τ -trait, we evaluate the task success performance using the pass^k metric (Yao et al., 2024). It is defined as the probability that all of the k independently sampled outputs successfully complete the task, averaged across all tasks. Specifically, if a task is run for n independent trials and c of those are successful (i.e., have a correct result with reward $r = 1$), then pass^k can be computed using the following formula:

$$\text{pass}^k = \mathbb{E}_{\text{task}} \left[\frac{\binom{c}{k}}{\binom{n}{k}} \right]$$

ACEBench ACEBench is a tool used for benchmarking centered on function-call correctness across broad everyday domains, built in two parallel languages (English and Chinese) with a final size of 2,000 annotated entries. It includes a large synthetic API collection with 4,538 APIs spanning 8 major domains and 68 sub-domains (including technology, finance, entertainment, society, health, culture, environment, and others). From 50 agent-based tasks, we have mainly focused on 30 agent

Method	End-to-End Acc	Process Acc
<i>Qwen3-4B-Instruct</i>		
Base	10.0%	41.6%
ReAct	10.0%	51.3%
FAMA-Base (our)	16.7%	45.1%
FAMA-ReAct (our)	20.0%	53.9%
<hr/>		
<i>Qwen3-14B</i>		
Base	16.7%	40.8%
ReAct	20.0%	61.3%
FAMA-Base (our)	23.3%	50.0%
FAMA-ReAct (our)	23.3%	58.4%
<hr/>		
<i>Qwen3-32B</i>		
Base	23.3%	46.5%
ReAct	13.3%	50.2%
FAMA-Base (our)	23.3%	50.0%
FAMA-ReAct (our)	30.0%	59.6%
<hr/>		
<i>Qwen2.5-72B-Instruct</i>		
Base	23.3%	45.1%
ReAct	36.7%	57.6%
FAMA-Base (our)	50.0%	70.2%
FAMA-ReAct (our)	43.3%	65.2%

Table 3: Comparison of agentic approaches across different open-source agents on ACEBench.

multi-turn tasks for our work, discarding the other 20 multi-step tasks.

For ACEBench, End-to-End Accuracy measures the degree of correspondence between the predicted instance attributes and the ground truth. When all attributes align perfectly, the accuracy equals 1; any discrepancy results in an accuracy of 0. Process Accuracy quantifies how closely the actual sequence of function calls aligns with the expected (ideal) sequence. This metric is defined as $\frac{n}{m}$, where m denotes the total number of steps in the ideal function call sequence, and n indicates the number of steps where the actual process matches the ideal one.

C Analysis of the τ -trait Results

To demonstrate the generalizability of the FAMA framework, we evaluate it on τ -trait, which includes four domains: Retail, Airline, Telehealth, and Telecom. Since we have already analyzed performance on the Airline and Retail domains in τ -bench, we focus our evaluation on the Telehealth and Telecom domains.

Similar to τ -bench, Domain Constraint Violations (DCV) and Contextual Misinterpretation (CM) are identified as critical error sources by the orchestrator agent, as shown in Figure 12. However, unlike Tau-Bench, memory-related issues are more prominent across Tau-Trait domains, with

Incomplete Fulfillment (IFU) emerging as the primary cause of task failures. As illustrated in Figure 13, the mitigation agent frequently recommends the Memory module for a larger number of models. Consequently, using only the Memory module significantly improves the performance of open-source models and, in some cases, yields better results than combinations that also include the Domain Constraints Extractor (DCE) agent.

The results in Table 4 show that FAMA consistently outperforms IRMA across both the Telecom and Telehealth domains on pass^k, for $k = 1, \dots, 5$. While certain methods, such as FC, outperform FAMA in isolated cases, FC does not demonstrate consistent performance across domains and exhibits substantial variability depending on the domain and model. These results further highlight the robustness and generalizability of the FAMA framework across diverse settings.

D Analysis of the ACEBench Results

ACEBench is another benchmark used to evaluate the FAMA framework. Although ACEBench supports multiple evaluation settings, we focus on the multi-turn agent configuration, where an active user agent interacts with a tool-calling agent. We compare FAMA against the Base Agent defined in ACEBench and the ReAct framework.

The results in Figure 14 illustrate the distribution of failure causes on ACEBench as identified by the orchestrator agent. As shown in the figure, all evaluated models suffer from significant Contextual Misinterpretation (CM) issues, manifested as incomplete task execution, misinterpretation during problem solving, and incorrect interactions with the environment. To address these issues, the mitigation agent consistently recommends the inclusion of Planner and Verifier agents, which are well-suited to correcting such failures as shown in Figure 15.

The results reported in Table 3 show that augmenting ReAct and the Base Agent with FAMA leads to consistent performance improvements over other baselines, even with the addition of only two auxiliary agents. In the following section, we present a qualitative example illustrating how FAMA mitigates these issues and successfully completes a task.

D.1 Example

In this example, we use Qwen2.5-72B-Instruct as both the tool-calling agent and the user agent. The example corresponds to a message-sending task from ACE-Bench. We compare the interaction produced by the Base Agent, as defined in ACEBench, with the interaction generated using the FAMA framework.

Example: Baseline

User: Can you help me send a message to Frank? The content is: 'Are you free tomorrow to have lunch with me?'

Assistant: send_message(sender_name='user', receiver_name='Frank', message='Are you free tomorrow to have lunch with me?')

Tool: "status": false, "message": "Wi-Fi is turned off, cannot send messages at this time"

Assistant: turn_on_wifi()

Tool: "status": true, "message": "Wi-Fi has been turned on"

Assistant: send_message(sender_name='user', receiver_name='Frank', message='Are you free tomorrow to have lunch with me?')

Tool: "status": false, "message": "Inbox capacity is full. You need to ask the user which message to delete."

Assistant: Memory is full, which message would you like to delete?

User: Please delete the oldest message.

Assistant: get_earliest_message_id()

Tool: "status": true, "message": "The earliest message ID is 3", "message_id": 3

Assistant: delete_message(message_id=3)

Tool: "status": true, "message": "Message ID 3 has been successfully deleted."

Assistant: send_message(sender_name='user', receiver_name='Frank', message='Are you free tomorrow to have lunch with me?')

Tool: "status": false, "message": "Sender or receiver does not exist"

Assistant: Could you please confirm the correct name of the recipient?

User: The recipient's name is indeed Frank. Could you proceed with sending the message?

Assistant: send_message(...) → **Repeated failures...**

User: The recipient's name is correct... Here is his phone number: 123-456-7890.

Assistant: send_message(..., phone_number='123-456-7890')

User: Here is his email address frank@example.com

Assistant: send_message(email='frank@example.com')

Result: **Task Failed** – Agent loops without resolving the sender name issue.

FAMA Framework

User: Can you help me send a message to Frank? The content is: 'Are you free tomorrow to have lunch with me?'

Planner:

1. Ensure device is logged in: login_device()

Method	τ -Telehealth					τ -Telecom				
	Pass^1	Pass^2	Pass^3	Pass^4	Pass^5	Pass^1	Pass^2	Pass^3	Pass^4	Pass^5
Qwen3-4B-Instruct										
ReAct	22.99%	16.00%	12.50%	11.00%	10.00%	15.55%	8.33%	6.11%	5.55%	5.55%
FC	34.00%	27.00%	24.00%	22.00%	20.00%	37.00%	31.00%	27.00%	24.00%	21.00%
IRMA	22.99%	14.50%	11.00%	10.00%	10.00%	<u>38.00%</u>	26.00%	21.00%	<u>18.00%</u>	16.00%
FAMA (our)	<u>32.00%</u>	<u>24.00%</u>	<u>20.00%</u>	<u>18.00%</u>	<u>17.00%</u>	39.00%	<u>28.00%</u>	<u>22.00%</u>	<u>18.00%</u>	<u>17.00%</u>
Qwen3-14B										
ReAct	44.00%	43.00%	42.00%	41.00%	40.00%	15.55%	11.11%	11.11%	11.11%	11.11%
FC	39.00%	38.00%	37.00%	36.00%	35.00%	<u>38.00%</u>	<u>33.33%</u>	<u>30.00%</u>	<u>28.00%</u>	<u>27.00%</u>
IRMA	39.99%	27.99%	24.50%	22.00%	20.00%	36.66%	23.30%	16.66%	13.33%	11.11%
FAMA (our)	<u>42.00%</u>	<u>41.00%</u>	<u>40.00%</u>	<u>40.00%</u>	40.00%	43.00%	35.50%	31.00%	29.00%	28.00%
Qwen3-32B										
ReAct	43.00%	37.00%	34.00%	32.00%	30.00%	24.44%	20.00%	18.88%	17.77%	16.66%
FC	42.00%	39.00%	37.00%	<u>35.00%</u>	35.00%	32.00%	25.00%	21.00%	18.00%	16.00%
IRMA	30.00%	24.50%	20.00%	18.00%	15.00%	40.00%	28.00%	24.44%	23.33%	22.20%
FAMA (our)	45.00%	<u>37.00%</u>	<u>35.00%</u>	35.00%	35.00%	<u>37.77%</u>	31.66%	30.00%	28.88%	27.77%
Qwen2.5-72B-Instruct										
ReAct	43.00%	39.00%	<u>38.00%</u>	<u>35.00%</u>	<u>34.00%</u>	<u>51.11%</u>	<u>39.44%</u>	<u>32.22%</u>	26.66%	22.22%
FC	41.00%	36.00%	32.00%	31.00%	30.00%	44.00%	35.00%	31.00%	28.00%	<u>27.00%</u>
IRMA	31.00%	28.50%	27.00%	26.00%	25.00%	44.44%	30.00%	23.88%	22.22%	22.22%
FAMA (our)	<u>42.00%</u>	40.00%	40.00%	40.00%	40.00%	52.00%	44.00%	41.10%	40.00%	39.00%

Table 4: Comparison of different open-source agents on τ -trait across five runs. Note that Qwen2.5-72B-Instruct is used as the model backbone of the sub-agents of IRMA and FAMA agentic frameworks.

2. Send message to Frank
3. Confirm with user about reminder
4. Finish conversation
Assistant: login_device()
Verifier: **PASS**; Correctly initiated device login.
Tool: "status": true, "message": "Device has been logged in"
Assistant: send_message(sender_name='User', receiver_name='Frank', message='...')
Verifier: **PASS**; Correctly sent the message as per the plan.
Tool: "status": false, "message": "Wi-Fi is turned off..."
Assistant: turn_on_wifi()
Verifier: **PASS**; Correctly identified the need to turn on Wi-Fi.
Tool: "status": true, "message": "Wi-Fi has been turned on"
Assistant: send_message(...)
Verifier: **RECHECK**; Wi-Fi was turned on but message was not sent; fix: retry send_message
Tool: "status": false, "message": "Inbox capacity is full..."
Assistant: Memory is full, which message would you like to delete?
Verifier: **PASS**; Correctly identified the need to delete a message.
User: Please delete the oldest message.
Planner:
1. get_earliest_message_id()
2. delete_message(message_id=EARLIEST MESSAGE ID)
3. Confirm deletion
Assistant: get_earliest_message_id()
Verifier: **PASS**; Correctly identified need to get earliest message ID.

Tool: message_id: 3
Assistant: delete_message(message_id=3)
Verifier: **PASS**; Correctly deletes the oldest message as requested.
Tool: "status": true, "message": "Message ID 3 deleted."
User: Sure, let's try again.
Assistant: send_message(sender_name='User', receiver_name='Frank', ...)
Verifier: **PASS**; Correctly attempting to send message after clearing memory.
Tool: "status": false, "message": "Sender or receiver does not exist"
Assistant: Could you please confirm the correct names for sender and receiver?
Verifier: **RECHECK**; Sender or receiver does not exist; fix: Ask user to confirm names.
User: The sender is Grace, and the receiver is Frank.
Planner: 1. Ensure grace is logged in
2. Send message from Grace to Frank
Assistant: send_message(sender_name='Grace', receiver_name='Frank', message='Are you free tomorrow to have lunch with me?')
Verifier: **PASS**; Correctly updated sender name and resent message.
Tool: "status": true, "message": "Message successfully sent to Frank."
Assistant: The message has been successfully sent to Frank. Is there anything else you need help with?
Verifier: **PASS**; Correctly sent the message and confirmed success.
User: finish conversation.
Planner:
1. Ask the user if they have any final tasks or requests before finishing the conversation.
2. If the user has no further tasks or requests, respond

with finish conversation to end the dialogue.

Result: Task Completed Successfully with Full Verification

The above example demonstrates that incorporating only a Planner and a Verifier agent, without additional auxiliary agents, is sufficient to successfully solve a tool-calling task.

E Error Categories and Causes of the Failures

In this section, we first describe the error categories and then elaborate on the underlying causes of failure associated with each category. The definition of each error category is provided below.

Domain Policy Violation: This error refers to a failure mode in which a tool-using agent issues an action or tool invocation that contravenes the explicit operational rules, constraints, or policies of the simulated domain defined by the environment. These policies govern which actions are permissible under specific states and preconditions. A violation occurs when the agent ignores, misinterprets, or incorrectly applies these domain rules, such as invoking a return operation that is disallowed by the retail domain’s return policy in τ -bench, resulting in an invalid state transition. Because the environment enforces domain policies strictly, such violations cause the task execution trajectory to terminate unsuccessfully, even if the agent’s intent aligns with the high-level task objective.

Incorrect Retrieval from Complex Tool Outputs: refers to a failure mode in which a tool-using agent fails to correctly extract relevant information from prior tool responses stored in its context. This error typically arises when tool outputs are large, unstructured, or weakly structured (e.g., verbose logs, nested text fields, or mixed natural language and metadata), making it difficult for the agent to reliably identify the correct entities, values, or state variables required for subsequent actions. As a result, the agent conditions later tool calls or reasoning steps on incorrect or incomplete information, leading to invalid decisions or task failure despite the necessary information being present in the environment output.

Contextual Misinterpretation and Hallucination: refers to a failure mode in which a tool-using agent incorrectly infers the user’s underlying intent due to nuanced, ambiguous, or adversarial

natural language instructions. Instead of grounding its interpretation in the domain’s available operations and constraints, the agent over-literalizes phrasing, overlooks implied intent, or hallucinates an incorrect task formulation. This results in the selection and invocation of an inappropriate tool, such as executing a return operation when the user implicitly intends an exchange, despite the correct intent being inferable from contextual or domain knowledge.

Incomplete Fulfillment or Early Stopping: refers to a failure mode in which a tool-using agent does not fully satisfy all components of the user’s task requirements before terminating or effectively abandoning the interaction. This error commonly arises in long-horizon, multi-turn trajectories where the agent must track multiple obligations across conversational turns while operating over large, unstructured tool outputs. As the interaction progresses, the agent may lose track of pending sub-goals, become over-focused on intermediate tool calls, or prematurely conclude the task, resulting in unmet user requests despite partial progress.

To enable judgment agents to detect task failures, a detailed list of failure causes is required. Accordingly, we analyze the trajectories produced by baseline methods and provide a comprehensive list of failure causes for each error category, as described below.

Cause of Domain Policy Violation

1. The assistant performs an action that the policy explicitly forbids
2. The assistant skips a step that the policy explicitly requires
3. The assistant proceeds without meeting conditions the policy requires
4. The assistant makes commitments that contradict policy rules
5. The assistant uses data/arguments not provided by user or tools (fabrication)

Cause of Wrong Retrieval from Complex Tool Outputs

1. The agent states information that differs from what the tool output actually contains
2. The agent miscounts items or provides wrong totals from tool outputs
3. The agent selects wrong items when correct options exist in tool output
4. The agent misses relevant data in tool output that would change the action
5. The agent attributes data to the wrong entity when

multiple exist in output

6. The agent uses wrong values (IDs, prices, dates, etc.) from tool output

Cause of Incomplete Fulfillment or Early Stopping

The user may have multiple requirements, questions, or actions they need completed. This error occurs when the agent:

1. Addresses only some parts of a multi-part request
2. Stops after encountering a difficulty without trying alternatives
3. Receives confirmation to proceed but never executes the action
4. Terminates the conversation with unfinished tasks

Cause of Contextual Misinterpretation and Hallucination

The user’s request may have specific intent, context, or nuances that the agent must correctly understand to take appropriate actions. This error occurs when the agent:

1. Understands the words but misses the actual intent
2. Confuses which entity/item/order the user is referring to
3. Misapplies criteria or constraints the user specified
4. Takes an action that doesn’t match what the user actually wanted

Model	Best Agent Configuration	
	Airline	Retail
Qwen3-4B-Instruct	DCE and Memory (K=6)	DCE and Memory (K=6)
Qwen3-14B	DCE and Memory (K=2)	Memory (K=6)
Qwen3-32B	DCE and Memory (K=2)	DCE and Memory (K=6)
Qwen2.5-72B-Instruct	DCE and Memory (K=4)	DCE and Memory (K=2)

Table 5: The different agents used in the FAMA framework for τ -bench.

Model	Best Agent Configuration	
	Telehealth	Telecom
Qwen3-4B-Instruct	DCE and Memory (K=2)	Memory (K=6)
Qwen3-14B	Memory (K=2)	DCE and Memory (K=2)
Qwen3-32B	Memory (K=6)	Memory (K=6)
Qwen2.5-72B-Instruct	Memory (K=4)	DCE and Memory (K=6)

Table 6: The different agents used in the FAMA framework for τ -trait.

F Optimal Components of FAMA

In this section, we describe the auxiliary agents and modules commonly used in static multi-agent frameworks and present, for each model, the final set of components selected by FAMA that achieves the best performance reported in Tables 1, 4, and 3.

For τ -bench and τ -trait, we assume the availability of the Memory module, the Domain Constraints Extractor (DCE) Agent, the Tool Output Reformulation (TOR) Agent, and the Tool Suggestion (TS) Agent. For ACE-Bench, we additionally include a Planner Agent and a Verifier Agent, as the base agent in ACE-Bench is unable to generate free-text and can only invoke tools with their input arguments. The final set of agents used in the FAMA framework is presented in Tables 5, 6, and 7.

Model	Best Agent Configuration
Qwen3-4B-Instruct	Planner and Verifier
Qwen3-14B	Planner and Verifier
Qwen3-32B	Planner and Verifier
Qwen2.5-72B-Instruct	Planner

Table 7: The different agents used in the FAMA framework for ACE-Bench.

G Comparison with Self-Reflection

As part of the rebuttal, we also implement and compare against a self-reflection (SR) baseline, inspired by approaches such as Self-RAG and “System 2”-style prompting (Madaan et al., 2023; Shinn et al., 2023). This baseline augments the base tool-use agent with an explicit reflection and correction phase, where intermediate outputs are analyzed and iteratively revised before producing the final response.

We evaluate this SR baseline alongside FAMA and ReAct across multiple models in τ -bench. Our results (Tables 9-12) show that FAMA consistently outperforms the SR baseline, suggesting that structured failure analysis combined with targeted mitigation is more effective than relying solely on generic reflection and revision mechanisms in our setting.

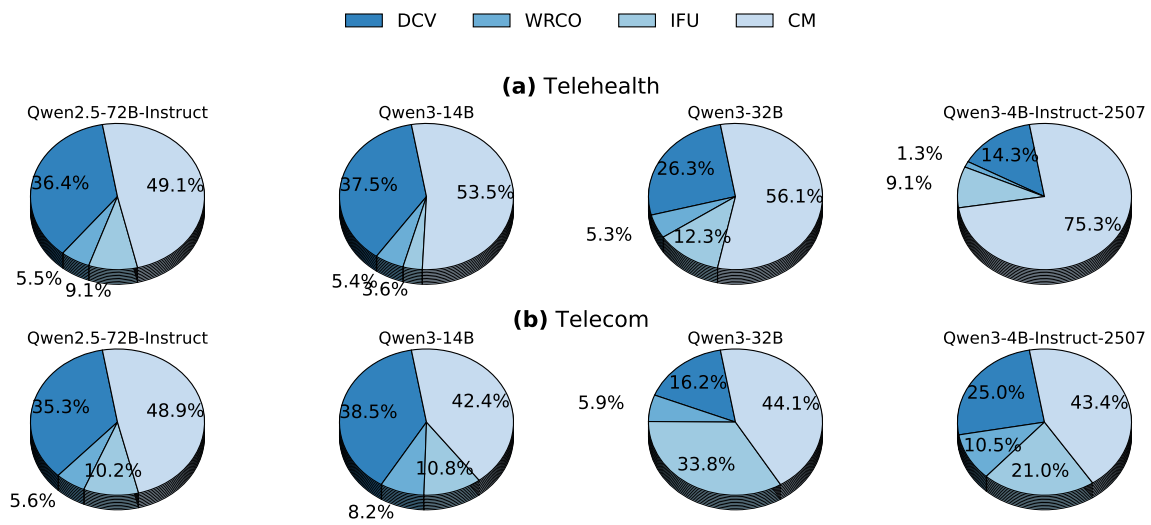


Figure 12: Statistical comparison of error categories across open-source LLMs in τ -trait environment, as identified by the Orchestrator Agent.

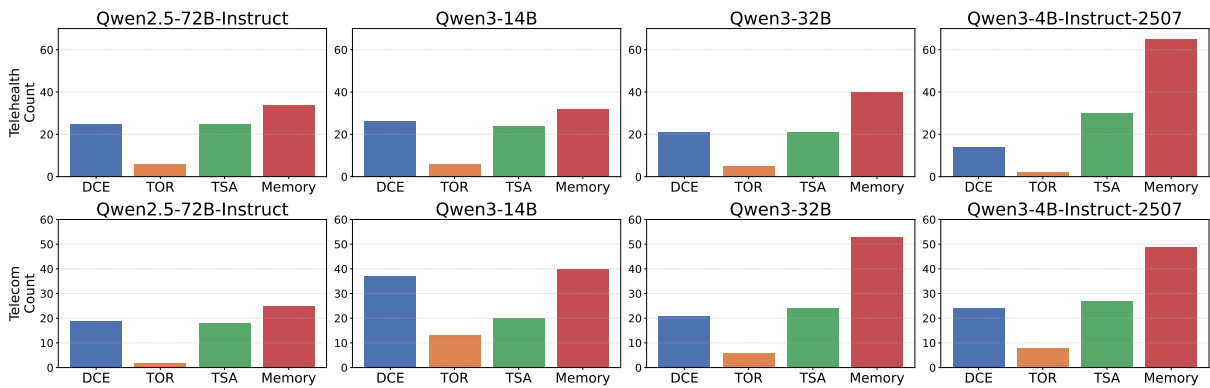


Figure 13: Statistics of recommendations produced by the Mitigation Agent across different open source agents.

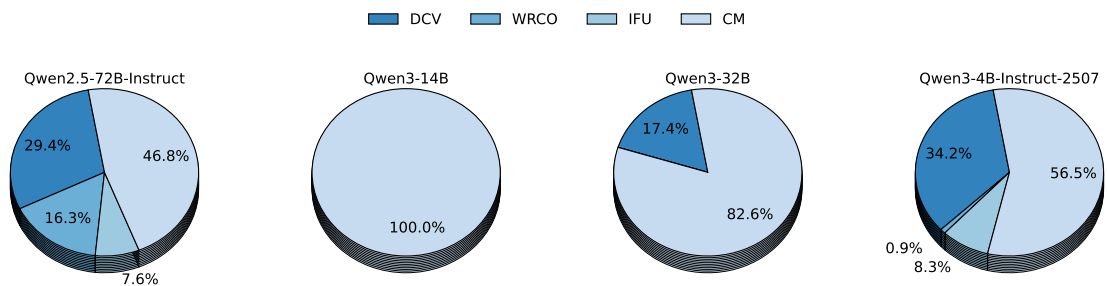


Figure 14: Statistical comparison of error categories across open-source LLMs in ACEBench, as identified by the Orchestrator Agent.

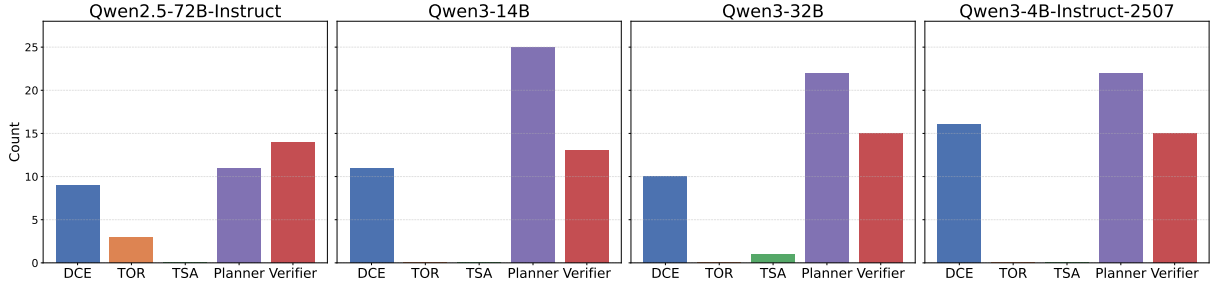


Figure 15: Statistics of recommendations produced by the Mitigation Agent across different open source agents for ACEBench.

Method	τ -airline	τ -retail
Q3-14B-Think-ReAct	0	0
Q3-14B-Think-FAMA	7	3
Q3-14B-Think-IRMA	8	6
Q3-32B-Think-ReAct	0	0
Q3-32B-Think-FAMA	12	20
Q3-32B-Think-IRMA	13	23

Table 8: Frequency of context window overflows for thinking model variants. Values indicate the number of task failures (measured for all tasks across 5 runs) where the maximum token budget was exceeded. Q3 denotes the Qwen3 model family.

τ -bench-retail					
Model	Pass ¹	Pass ²	Pass ³	Pass ⁴	Pass ⁵
ReAct	43.47	32.26	26.69	23.13	20.86
FAMA	44.173	34.26	30.26	28.17	26.95
SR	42.9	30.7	25	21.2	18.2

τ -bench-airline					
Model	Pass ¹	Pass ²	Pass ³	Pass ⁴	Pass ⁵
ReAct	24.4	18.79	15.6	12.8	10
FAMA	29.2	21.2	18.8	18	18
SR	28	20.4	17.4	15.6	14

Table 9: Pass^k results for Qwen2.5-72B-Instruct on τ -bench datasets comparing Self-Reflection performance against FAMA and ReAct.

τ -bench-retail					
Model	Pass ¹	Pass ²	Pass ³	Pass ⁴	Pass ⁵
ReAct	38	25	18.8	15.3	10
FAMA	40.5	26.9	19.9	15.3	12.2
SR	26.7	15.6	11.3	9.2	7.8

τ -bench-airline					
Model	Pass ¹	Pass ²	Pass ³	Pass ⁴	Pass ⁵
ReAct	30.4	20	16.2	14.8	14
FAMA	26.8	20	18.4	18	18
SR	25.2	17	13.2	10.4	8

Table 10: Pass^k results for Qwen3-32B on τ -bench.

τ -bench-retail					
Model	Pass ¹	Pass ²	Pass ³	Pass ⁴	Pass ⁵
ReAct	25.2	17.8	14.7	13.2	12.1
FAMA	37.9	25.7	19.7	16.3	14.7
SR	32.1	19.3	14.69	12.52	11.3

τ -bench-airline					
Model	Pass ¹	Pass ²	Pass ³	Pass ⁴	Pass ⁵
ReAct	18.4	12	9.4	8.4	8
FAMA	36.8	23.79	18.2	16.6	16
SR	21.2	15.2	13.3	12.4	12

Table 11: Pass^k results for Qwen3-14B on τ -bench.

τ -bench-retail					
Model	Pass ¹	Pass ²	Pass ³	Pass ⁴	Pass ⁵
ReAct	17.22	12.35	10.61	9.57	8.7
FAMA	34.6	24.1	19.3	16.3	13.9
SR	31.3	19.3	14.34	11.82	10.43

τ -bench-airline					
Model	Pass ¹	Pass ²	Pass ³	Pass ⁴	Pass ⁵
ReAct	32	28	26.8	26.4	26
FAMA	37.6	32	28.3	26.7	26
SR	33.2	23.9	19.6	16.39	14

Table 12: Pass^k results for Qwen3-4B-Instruct-2507 on τ -bench.