Personal Agents Need Deployment Context, Not More Training

Tapan Chugh [1] **Vinamra Agarwal** [1] **Parth Kotwal** [1] **Vidushi Singh** [1] **Vrinda Inani** [1] **Huanzhi Mao** [2]
Shishir G. Patil [2] **Arvind Krishnamurthy** [1] **Ratul Mahajan** [1]
[1]University of Washington [2]University of California, Berkeley

Abstract

State-of-the-art LLMs can support arbitrary function calls with in-context learning. However, practical improvements driven by recent advancements in post-training techniques do not generalize to **personal agents**—agents which act on a user's personal data and services on their behalf. We conduct a *grounded theory* analysis of 500+ failure logs across 4 frontier models and 3 benchmarks, and present **AFT**: an **A**gent **F**ailure **T**axonomy. Analyzing **expert annotated** failure logs reveal that models do not fail from inadequate reasoning or function invocation capabilities but struggle with understanding *task intent* and often lack *tacit domain knowledge* needed to complete them. This paper argues that although training improves basic LLM capabilities, scaling existing training paradigms is insufficient to address these context gaps. Instead agents must acquire the ability to **identify** context gaps, **involve** humans strategically, and **internalize** acquired knowledge over time. We discuss considerations for realizing such systems and highlight open research questions.

1 Introduction

> *"On two occasions I have been asked, 'Pray, Mr. Babbage, if you put into the machine wrong figures, will the right answers come out?' I am not able rightly to apprehend the kind of confusion of ideas that could provoke such a question."*
>
> — Charles Babbage

Consider an AI assistant connected to a user's Internet accounts that enables offloading tasks such as: (1) Check if any unread emails require an immediate response. (2) Create a pull request for these code changes to close the issue. (3) Why did Alice change this document last week?

Even though Large Language Models (LLM) advancements have enabled broad adoption of AI agents for web-search (e.g., Deep Research (Google DeepMind)) and software programming (e.g., Cursor (Anysphere), Claude Code (Anthropic, b)), such *personal agents*—AI systems which act on a user's personal data and services on their behalf—face more challenging requirements Figure 1; for instance, prior research (Ai et al., 2017; Kim et al., 2017) identified that when searching over personal data, users frequently seek to retrieve and act upon specific previously seen content (e.g., a particular email), whereas web-search users often conduct exploratory queries and accept approximate results. Similarly, unlike coding, where agents can make mistakes, run tests, and fix errors (Yang et al., 2024), actions like sending emails or rescheduling meetings offer little margin for error. Thus, although community uptake of open standards, e.g., Model Context Protocol (MCP) (Anthropic, a) has enabled the development of many interesting prototypes (Luo et al., 2025b), production deployments still struggle with limited accuracy. (Pan et al., 2025; NANDA, 2025)

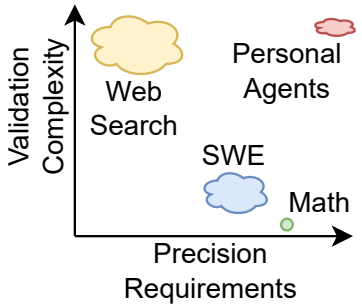

Figure 1: Comparing different types of agents along two axes; personal agents face stricter precision requirements, while it is harder to validate proposed solutions and fix them.

Further, while accuracy benefits for coding and search applications have been enabled by incorporating workload-specific data directly into the training pipeline, their implications for personal agents are unclear. Although representative workloads are lacking, we evaluated state-of-the-art LLMs on 3 popular function calling benchmarks which include tasks with similar requirements as in Figure 1 : Berkeley Function Calling Leaderboard (BFCL) (Patil et al.), AppWorld (Trivedi et al., 2024), and MCP-Universe (Luo et al., 2025b). Our results, presented in Figure 2 suggest significant accuracy headroom, while longitudinal analysis in Figure 3 suggests that although LLM capabilities have improved rapidly on SWE-bench (Jimenez et al., 2023), which

Model	BFCL	AppWorld	MCP-U
GPT-5	45.84	87.8	57.1
Sonnet 4.5	68.4	91.8	50.0
Kimi K2	65.5	34.7	46.4
Qwen3-32B	58.5	27.9	17.9

Figure 2: Evaluating accuracy of present LLMs on 3 benchmarks; see §2 for details.

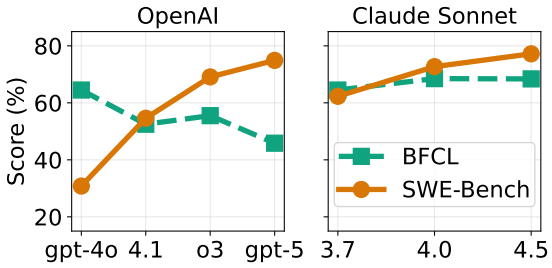

Figure 3: SWE-Bench and BFCL accuracy changes over time.

evaluates performance on coding tasks, the performance on BFCL has grown marginally and sometimes varied non-monotonically across model generations.

Although LLM capabilities are evolving rapidly, the economic and societal significance of the workload necessitates a detailed examination. While function calling errors have been studied recently for older models (Winston & Just, 2025), specific LLM applications (Song et al., 2023; Bai et al., 2024) and agentic systems (Cemri et al., 2025; Zhang et al., 2025c), personal agents have not been studied scientifically, to the best of our knowledge.

Therefore, we conduct a **grounded theory** (Glaser & Strauss, 2017) analysis, wherein instead of testing a specific hypothesis, we collect and analyze a dataset of failure logs from 4 frontier LLMs on 3 benchmarks to understand *why personal agents fail*. We manually examine and annotate 500+ failure logs to prepare **AFT**: an **A**gent **F**ailure **T**axonomy which classifies failed trajectories into fine-grained categories. Our findings indicate that:

- **Intent understanding** presents a persistent challenge: models either perform actions assuming a specific interpretation, leading to mismatched expectations, or repeatedly stop for user clarifications.
- **Missing tacit knowledge** seems responsible for *reasoning* models inability to orchestrate complex tasks; GPT-5 and Sonnet 4.5 frequently completed such tasks when sufficient context was available, while others continued to struggle (Kate et al., 2025) to complete tasks across many turns using large # of tools.
- **Outcomes might worsen** from excessive post-training; we observed that models repeatedly made similar errors—consistent with the hypothesis that excessive RLVR can reduce output diversity (Yue et al., 2025; Wu & Choi, 2025; Zhao et al., 2025)—and fabricated results more frequently in real-world setups than in mock environments likely due to generalization gaps.

Although ongoing research (He et al., 2025; Gunjal et al., 2025; Shao et al., 2025; Yu et al., 2025) and economic investments (bes) might eventually help, it is worth noting that interpreting intent accurately has challenged AI assistants for over 30 years (Etzioni & Weld, 1994), while generalization gaps between simulated and real environments are also not unique and remain endemic across multiple RL applications (Xie et al., 2024; Aljalbout et al., 2025). Thus, although these findings might seem obvious in retrospect, they raise crucial concerns regarding our community's commonly held hypotheses for how to improve personal agents: (i) Scaling the training for general *reasoning* capabilities might lead to *emergent* behavior that improves performance for specific applications. (ii) Even if trained models do not generalize, present (and future) training techniques can be applied to improve accuracy for arbitrary applications. Although training indeed seems to improve basic reasoning and execution capabilities, our results indicate that those are unlikely the bottleneck, and while significant research is ongoing on (ii), progress beyond *verifiable* domains, where agents can *self-explore* in *mock environments*, remains limited. Given the unresolved theoretical and practical challenges, it is worth critically examining if alternative approaches might work.

Our Position: We argue that scaling existing training paradigms will NOT lead to better personal agents. Instead, we propose that such systems require 3 basic mechanisms: (i) *identify* missing context, (ii) *involve* humans or (or other agents) to acquire missing context, and (iii) *internalize* the learned experience to inform future behavior appropriately. We present a high-level overview of how such a system can be assembled, building on recent work which *learns* contextual affordances, e.g., playbooks (Zhang et al., 2025b), cheatsheets (Suzgun et al., 2025), etc., without updating the model parameters, and discuss open research questions and challenges in addressing them.

2 Why Personal Agents Fail?

To thoroughly understand agent failures, we conducted a grounded theory (Glaser & Strauss, 2017) analysis: rather than starting with a hypothesis and collecting evidence, we began by analyzing an initial dataset and identified the main categories of failures. Our analyzed dataset was generated by evaluating state-of-the-art models against multiple benchmarks; we collected the complete traces, including the output messages, tool call invocations and results, and organized them into detailed failure reports with information such as system state before the agent started, logs generated by the benchmark evaluators, ground truth solutions, etc.

Aspect	BFCL	AppW	MCP-U
Evaluators	Trajectory + State Match	Final State Unit Tests	Final State Unit Tests
Agent Tasks	LLM Generated (Personalized)	Curated Templates	Expert Curated
Function Defs	Custom APIs (Synthetic)	Mock Popular App. APIs	**Prod. GitHub MCP Server**
Input Data	Synthetic	Synthetic	**Live Data**

Table 4: Key qualitative aspects of analyzed benchmarks.

# Functions	BFCL	AppW	MCP-U
Total	129	473	93
Offered†	17–39 (28)	20–142 (95)	93
Used†	3–7 (6)	5–12 (8)	4–9 (6)
Calls †	3–7 (6)	5–244 (39)	6–34 (11)

Table 5: Quantifying differences in function calling benchmarks. Rows marked † present the range across test-cases with median values in parentheses; MCP-U has a fixed value.

Models: We evaluated both closed and open models[1]: (i) OpenAI GPT-5, (ii) Claude Sonnet 4.5, (iii) Kimi K2, and (iv) Qwen3 32B. For (i) and (ii), we used their default reasoning budgets; effects of varying reasoning budgets are discussed further in §2.3.

Workloads: In the absence of production data to analyze personal agents, we analyzed three function-calling benchmarks which stress models on tasks with high precision requirements and validation overheads: (i) Berkeley Function Calling Leaderboard (**BFCL**), (ii) AppWorld (**AppW**), and (iii) MCP-Universe (**MCP-U**). To provide comprehensive high-quality annotations, we selected a representative subset of test cases from each benchmark and annotated all the samples within that category: for BFCL, we used the *multi_turn_base* category; for AppWorld, their *train* set, and for MCP-Universe, their *repository management* category.

As Table 4 shows, we selected benchmarks that use high-quality evaluators that check correctness based on actual function execution: AppWorld and MCP-Universe define a set of *unit-tests* that evaluate changes, or lack thereof, on the final state, and BFCL additionally compares the trajectories against a golden *subset*, to ensure models indeed invoke functions rather than simply hallucinating or memorizing outputs, while also comparing the final state more comprehensively to ensure models do not introduce arbitrary side effects. We disregarded benchmarks which evaluate single function invocation and not multi-step tasks (Patil et al., 2024; Li et al., 2023), and those which use LLM-as-a-judge evaluators(e.g., (Bandi et al., 2025; Guo et al., 2024)), since they can be unreliable for agentic workloads (Luo et al., 2025b); while τ-bench (Yao et al., 2024) and τ^2-bench (Barres et al., 2025) do compare the final states, they use an LLM to simulate user behavior on incomplete responses, and the choice of this user-model can additionally confound the evaluation, and so we disregarded them as well.

Further, to ensure sufficient coverage under diverse settings, we selected benchmarks with distinct task and function profiles: (i) BFCL challenges model generalization in diverse settings using LLM-generated synthetic task instructions, which are personalized using PersonaHub (Ge et al., 2024). Also, their function definitions, e.g., for trading, ticket management, filesystem, etc., are unique and unlikely to be in popular pre-training corpora. (ii) AppWorld challenges models with complex detail-oriented tasks needing up to O(100) function calls (see: Table 5). Tasks are instantiated using different values for placeholders in expert-defined *scenario* templates with varying hardness, and require operating on popular applications and websites (e.g., Spotify, Venmo); in contrast to BFCL, pre-training corpora likely include significant background information about them. To ensure benchmark reproducibility, tasks execute mock APIs, e.g., CRUD (create, read, update, delete) for these services implemented by experts. (iii) MCP-Universe analyzes performance on live data using the production Github MCP-Server, with expert-curated, challenging tasks that could not be solved by models easily.

Grounded Theory: To create a comprehensive taxonomy of agent failures, we carefully examined each log to answer several questions: (1) Why did this test case fail the benchmark evaluation? (2) When the agent failed, did it complete the task or not? (3) If the agent completed the task and still failed, what caused the expectation mismatch? (4) Why wasn't the task completed? (5) Why did the agent select an incorrect function? and so on. We answered these questions—that emerged naturally during our investigation—by creating different category labels (*open coding* (Khandkar, 2009)) and followed *constant comparative analysis* (Glaser & Strauss, 2017) to update them.

Limitations: Our investigation found that a major fraction of test cases failed due to benchmark issues[2], rather than agent failures. We fixed the most frequent errors and repeated those tests, until additional tests did not yield new insights (*theoretical sampling* (Draucker et al., 2007)). Thus, although our taxonomy is comprehensive, the quantitative breakdowns in Figure 6 might change when the remaining $\sim$30% of non-agent errors are fixed and analyzed; more details on these errors are in §B.

AFT Categories: Overall, our investigation unearthed many different types of failures, defined in Figure 13, which likely arise from limitations across several stages, such as,

[1]We used the official providers APIs for all models

[2]We reported the issues and our suggested mitigations to the benchmark maintainers.

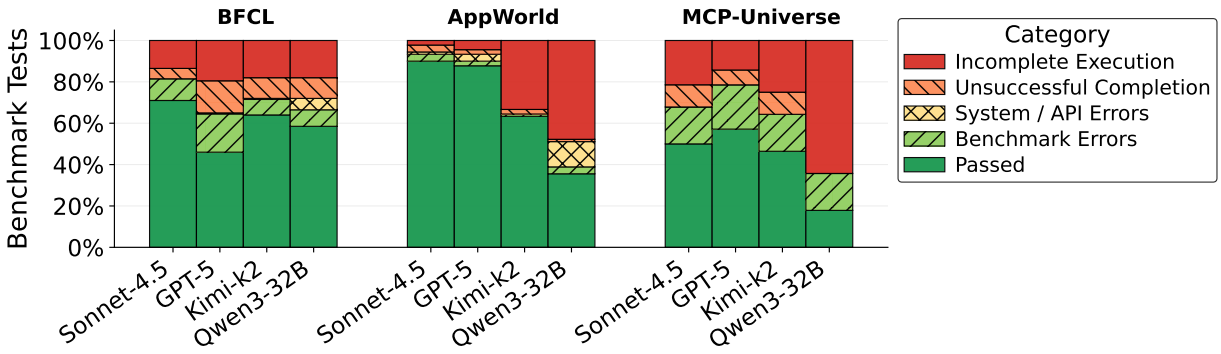

Figure 6: High-level breakdown of the analyzed test-cases into top-level AFT categories.

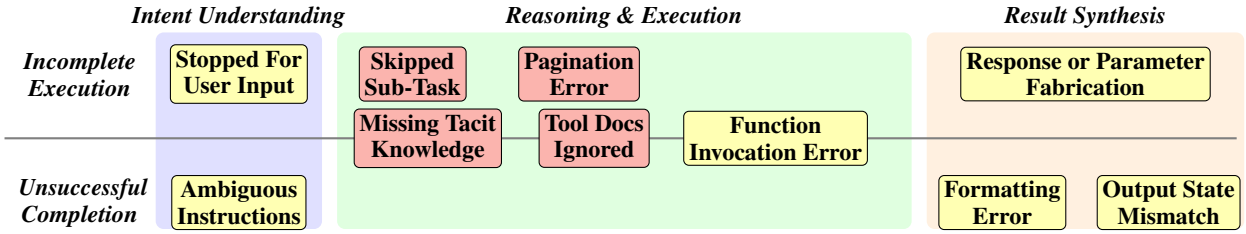

Figure 7: Agent failure categories organized by phase (horizontal) and completion status (vertical). Categories in the middle row (Missing Tacit Knowledge, Tool Docs Ignored, Function Invocation Error) can result in either incomplete execution or unsuccessful completion depending on whether the agent self-corrects (§A).

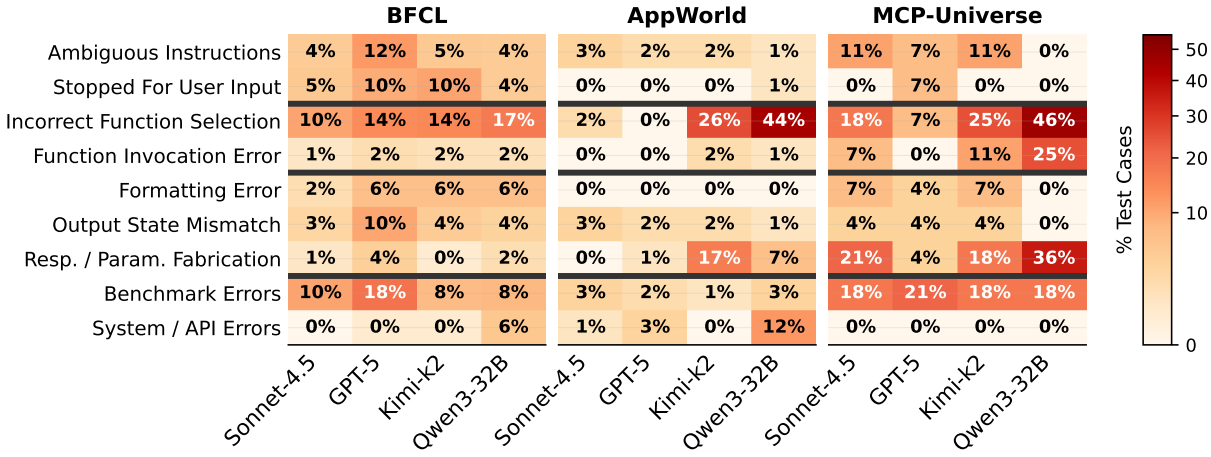

Figure 8: Prevalance of different types of errors across benchmarks; solid horizontal lines separate different error types—intent, reasoning, result, non-agent. Note each benchmark failure can belong to multiple sub-categories in AFT taxonomy.

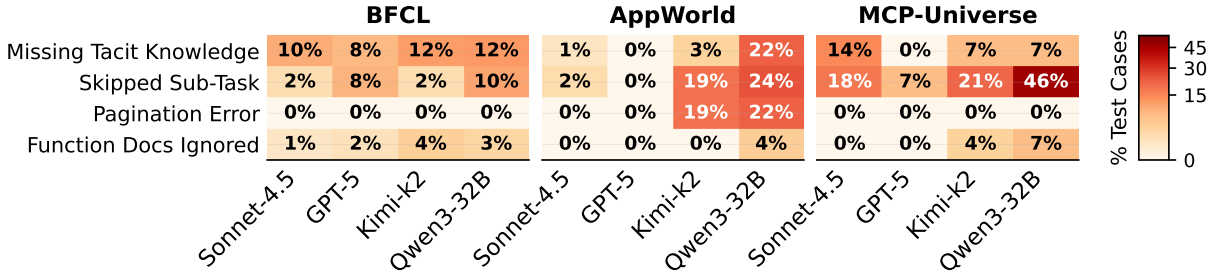

Figure 9: Prevalance of different types of incorrect function selection errors.

interpreting intent correctly, *reasoning & execution*, and *result generation/synthesis* (Figure 7). Notably, our investigation revealed that a single failed step, e.g., when invoking a specific function, or a missed step in the plan, was not necessarily indicative of end-to-end task completion; when provided helpful error messages, agents frequently attempted to self-correct their behavior. Thus, although our dataset only includes some categories for incomplete execution, such errors were also observed for unsuccessful completion cases, but not the primary reason why the agent failed.

The rest of this section describes the key findings from analyzing the expert-annotated dataset.

2.1 Intent Interpretation Remains Challenging

Our investigation identified two kinds of failures attributable to intent understanding: models either assume a specific interpretation leading to downstream errors causing unsuccessful completions from ambiguous instructions, or stopped for user input unnecessarily leading to incomplete executions; although the latter might be considered safer, it is severely limiting for the development of agents that complete complex tasks autonomously.

As Figure 6 shows, up to 15% of analyzed failures were unsuccessful completions—where models completed the provided instructions but had minor output state mismatches or formatting errors and did not exactly match benchmark expectations—rather than conceptual reasoning failures or hallucinations. This highlights the precise, detail-oriented nature of these workloads; unlike software engineering, where alternative valid solutions might be acceptable, and models can often run tests to identify mismatched expectations, similar affordances when drafting a document, creating a support ticket, etc., are unavailable.

Figure 8 shows that $\sim$10% of test cases were due to ambiguous instructions, where reasonable interpretations might differ. Notably, $\sim$17% of all BFCL test-cases were considered unsuccessful completions with ambiguous instructions at least once, while only $\sim$4% of BFCL test cases failed due to ambiguity for multiple models, indicating that implicit expectation misalignment, rather than explicit specification gaps, was a persistent challenge. Further, although incomplete executions predominantly arose due to incorrect function selection across all benchmarks, stopped for user input affected $\sim$10% of BFCL failures across models (Figure 8).

The differences in prevalence across benchmarks arise primarily due to their task curation processes and their evaluation prompts: BFCL explicitly attempts to stress intent understanding and has a separate test category which evaluates LLMs' ability to identify incomplete instructions; although their prompts encourage the model to stop for missing information, none of our analyzed test-cases actually required that. Further, BFCL tasks required generating free-form outputs, e.g., communication messages, support tickets, Twitter posts, etc., which led to more frequent formatting errors compared to other benchmarks that expected outputs in specific formats, e.g., CSV, Python code, etc.

2.2 Missing Tacit Knowledge Limits Task Completion

Although our analysis identified several reasons for agents failing to complete tasks (see Figure 13), and although incorrect function selection errors dominated across all models and benchmarks, the underlying causes differed: missing tacit knowledge was the prominent cause of failures (56-90%) across models on BFCL, and for Sonnet-4.5 failures on other benchmarks. (see Figure 9).

The differences between reasoning models (GPT-5 and Sonnet-4.5) and non-reasoning models (Kimi-K2 and Qwen3-32B) were more prominent on other benchmarks, with larger accuracy gaps (Figure 2) and different failure modes. As Figure 8 shows, for *non-reasoning* models, Skipped Sub-Task and Pagination Errors were the primary causes of Incorrect Function Selection on AppWorld and MCP-U, rather than Missing Tacit Knowledge. Note that we highlight Pagination Errors separately for AppWorld because, although their mock implementation was not realistic[3], their system prompt instructed models of such expectations, which were sufficient for GPT-5 and Sonnet-4.5; however, the performance of these models on more realistic pagination APIs is unclear. Nonetheless, since LLMs are evolving rapidly, we anticipate that Missing Tacit Knowledge, rather than Skipped Sub-Task or other limitations, will become the primary bottleneck for future reasoning-capable Kimi and Qwen models.[4]

2.3 Reasoning Helps Tasks With Sufficient Context

While previous reports (Kate et al., 2025; Paramanayakam et al., 2025; Gan & Sun, 2025) suggest that LLMs struggle to complete tasks requiring many turns, especially when offered more than 20–40 functions, our results indicate otherwise: GPT-5 and Sonnet-4.5 performed significantly better on AppWorld—which offers substantially more functions—than BFCL, which offers fewer functions and requires fewer calls per task (Table 5). Using synthetic ablation we find that

[3]Pagination refers to APIs splitting large response arrays across multiple pages. AppWorld's implementation did not provide any metadata (e.g., a next cursor) to inform the caller about the presence of additional data, which is standard practice. Instead, models were expected to check if the number of returned entries matches the page size, or always check subsequent pages for additional results.

[4]Although a reasoning variant of Kimi-K2 is already available, we were unable to study it similarly due to the labor-intensive nature of our analysis.

missing context and not task complexity is the bottleneck.

First, we validated that simply reducing the # functions offered does not affect AppWorld and MCP-Universe results: for AppWorld, we repeated the evaluation in *oracle* mode—which selects only the functions that are required for a task, and for MCP-Universe, we manually identified an oracle subset which has $\sim 5\times$ fewer functions than those exposed originally; in both cases, we saw minimal accuracy differences with GPT-5 and Sonnet 4.5.

Next, we confirmed that reasoning improvements help with task complexity, not missing context: evaluating GPT-5 with reduced reasoning effort led to reduced accuracy substantially for MCP-Universe and AppWorld, but varying the reasoning budgets for GPT-5 and Claude Sonnet 4.5 on BFCL did not affect the accuracy, as shown in Figure 10.

Lastly, we confirmed that adding missing context, e.g., missing tacit assumptions and hints to improve intent interpretation into the system prompt improves accuracy; we selected two subsets of failing test-cases from BFCL and provided them with relevant context identified during our expert annotation, which improved overall accuracy by $\sim 10\%$.

These findings indicate that improvements to general reasoning capabilities stemming from recent training strategies indeed help broadly with complex tasks. While we anticipate reasoning to improve in future models as well, context gaps are likely the primary bottleneck for better outcomes.

2.4 Generalization Can Worsen From Over-Training

When analyzing GPT-5 results on BFCL, we observed that when unsuccessful completions were not attributable to ambiguous instructions, certain functions and operations were highly indicative of misaligned expectations. For instance, for $\sim 50\%$ of the tests that were expected to create a support ticket, GPT-5 created the ticket but explicitly selected a different priority level than the default value, which was expected in the result. Similarly, for $\sim 95\%$ of the test cases that expected starting a car, the expected workflow required the model to press the brake pedal and start the engine, while GPT-5 always included an additional step to release the brake pedal after starting the engine—other models did not make this error, and previous versions of OpenAI models (e.g., GPT-4.1) also did not make such errors. Our observations align with concerns raised in prior studies that excessive RL training can reduce output diversity.

Further, we found that although Sonnet 4.5 performed better than GPT-5 on BFCL and AppWorld, it performed worse and frequently fabricated results on MCP-Universe, which evaluates models on live services (Figure 6; manual investigation revealed that when being tasked with copying a specific file from an existing Github repository, the model initially attempted to solve the task correctly by searching

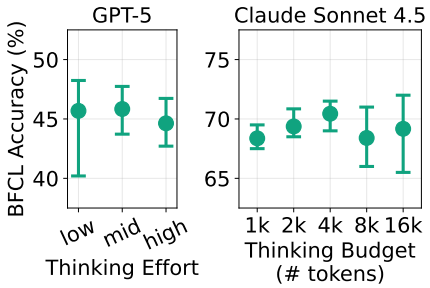

Figure 10: BFCL accuracy with varying thinking effort – graph shows the mean and range (max-min) over 5 trials.

for the file in the repository, but when it did not find it immediately—due to incorrect syntax for the search query—instead of attempting alternative searches, it created "a helpful file" from scratch instead, raising concerns regarding generalization in novel unexpected situations. We discuss additional concerns with scaling training to improve personal agents in §4.

3 Position: Deployment Context Acquisition

Given that current improvements in reasoning are insufficient for personal agents, we argue that future efforts must enable agents to acquire and apply context during deployment without updating model parameters. Unlike existing continual learning approaches (Ibrahim et al., 2024; Luo et al., 2025a), which aim to update parameters during deployment, we propose that deployment updates be handled entirely in-context, once initial training has bootstrapped sufficient capabilities. Unlike recent work (Zhang et al., 2025b; Suzgun et al., 2025), which also improves agents by learning context through self-exploration in verifiable mock environments, we advocate for learning from human feedback during deployment.

While building such agents might require some modeling changes, e.g., to ensure external feedback beyond internal knowledge is applied effectively, conceptually, such *active learning* (Prince, 2004) requires three core mechanisms: **identify** when context is missing, **involve** humans or external sources to acquire it, and **internalize** information to inform future behavior. Figure 11 illustrates such a system: at each step, the agent retrieves relevant experiences from prior interactions and uses that along with its internal reflection to generate the next action. Before executing, it evaluates if there is sufficient confidence to proceed autonomously; if not, a human (or another agent) is involved to provide feedback. Reflections gathered from past actions and external feedback are then internalized to improve future performance.

In the rest of this section, we discuss the design space of such systems, and open questions and challenges that must be addressed for further progress.

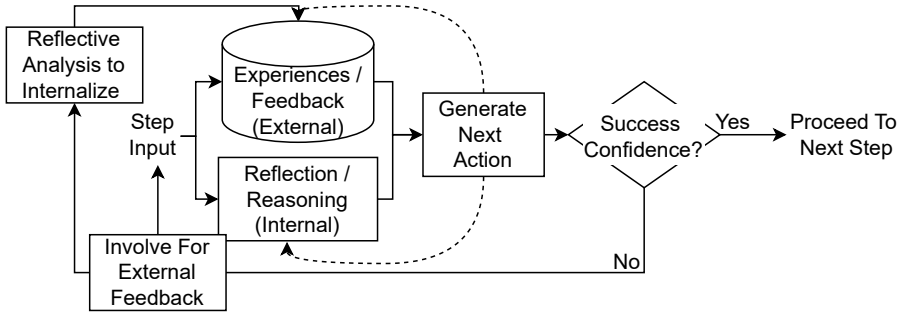

Figure 11: High-level overview of an agent that can identify, involve, and internalize context during deployment

Identify: Despite significant research attempting to improve LLMs uncertainty estimation (Kadavath et al., 2022; Kuhn et al., 2023; Lin et al., 2022; Xiong et al., 2024), our experiments (§2.1) indicate that when present models attempt to identify context gaps, the outcomes can be suboptimal: models either assume specific interpretations and proceed (leading to unsuccessful completions) or request clarification excessively (leading to incomplete executions). Since such *model-identified* involvement can be unreliable, *harness-identified* approaches work around this and get relevant context in specific places, e.g., by injecting relevant memories or cheatsheets when starting on a task, requiring human involvement after planning a complex task, etc.; although helpful, these might be limited for agents executing independently for up to hours at a time. In addition to potentially making *harness-identified* involvement more fine-grained, we argue that agentic systems consider an additional external *guardrail-identified* stage that evaluates confidence for the current step, and involves external feedback, either from humans or external context.

Such an approach is advantageous for several reasons: First, confidence evaluation is likely to be an easier problem than verification, and some false positives might be acceptable. Second, trainable guardrails can aggregate information across users to mitigate cold-start effects; while learned context or memories contain personal data that might not be shareable directly, a guardrail can analyze aggregate patterns and learn failure-specific signals; for instance, our analysis identified that models frequently failed when using specific functions, or were likely to fabricate files when previous searches did not yield successful results. Third, guardrails can analyze past outcomes and incorporate additional user and domain-specific features to improve predictions; while it might be possible to include these additional features into the LLM context directly, doing so can also hurt outcomes by confusing the models unnecessarily. Fourth, lightweight guardrails can be trained and updated frequently without modifying base model parameters, improving overall stability of the system without catastrophic forgetting.

Involve: When context gaps are identified, agents must acquire missing information efficiently, and different context gaps might require different sources. For harness and model-initiated involvement, recent research has evaluated several mechanisms to provide relevant context: (i) memory-based approaches, e.g., cheatsheets (Suzgun et al., 2025) and playbooks (Zhang et al., 2025b); (ii) contrastive learning providing positive and negative examples (Wu et al., 2024); and (iii) recursive language models (Zhang et al., 2025a) where agents can analyze previous traces directly. Although these have primarily been evaluated in specific domains, the mechanisms themselves are domain-agnostic. However, human involvement affordances will likely be domain-specific and require careful calibration: while human feedback is necessary to increase the overall information in the system, excessive involvement might hurt user satisfaction.

How should agents engage with humans meaningfully is a fundamental question in HCI (Horvitz, 1999; Bansal et al., 2021; Mozannar & Sontag, 2020); progress requires designing and evaluating interaction patterns, studying user mental models, and measuring effects on trust and satisfaction beyond task completion metrics. For agents, this raises a challenge: how can domain-agnostic techniques be developed and evaluated without overfitting to specific benchmarks?

Internalize: Determining how to extract insights from execution, how to retrieve them, and how to apply them later, offers several design choices which may require different strategies based on different context gaps. Additionally, our conversations with practitioners attempting to apply such strategies in production reveal recurring challenges arising from: (i) *marginal utility saturation*: systems use fixed prompts and workflows to extract information, so once the corresponding insights are captured and internalized, the *fixed* reflection and analysis process does not extract additional insights; thus, to continue *learning* during deployment, the reflection and context processing stage must adapt to what knowledge has already been acquired; and (ii) *sparse reward overfitting*: systems acquire feedback from a few samples and apply the same patterns broadly without sufficient contextual understanding, which degrades the overall response quality. Surprisingly, both failure modes remain difficult to reproduce in current bench-

marks. Mitigating these challenges and improving the evaluation infrastructure to reproduce these dynamics is necessary for progress on context-acquiring agents which can generalize broadly.

4 Alternative Views

This paper argues that the ability to acquire deployment context, rather than exclusively scaling training is critical for unlocking personal agents. In this section, we examine potential arguments against this position:

4.1 More RL Is All You Need

Our findings indicate that intent interpretation (§2.1) and missing tacit knowledge (§2.2) are the primary bottleneck to better agents.

Improving intent alignment for personal agents using RL is complicated since current techniques explicitly train on unambiguous and verifiable tasks. Even if alternatives arise, alignment using RL for complex long-horizon tasks (Falahati et al., 2025) is challenging as errors propagate downstream—a recurring phenomenon in other RL workloads (Ross et al., 2011), e.g. robotics (Aljalbout et al., 2025). Further, intent disambiguation will likely be user and context-specific; it is unclear whether any technique maximizing rewards over a broad population would suffice. Lastly, concerns regarding specific training might harm general capabilities (Lin et al., 2024) and generalization gaps between mock and real environments remain unresolved.

4.2 Accurate Environments Are All You need

There is a growing belief, aligned with the scaling paradigm, that high-fidelity RL environments are necessary and *sufficient* for personal agents. However, it is unclear if sufficient fidelity can be achieved reasonably: personal agents might require carefully mocking orders of magnitude more features—which is further exacerbated by the commonly employed agile development practices (Beck et al.), that prioritize rapid development velocity and frequent iteration based on continuous feedback, suggesting that developing realistic environments and curating representative workloads might require ongoing effort rather than a one-off upfront investment. Furthermore, as models improve, *reward hacking* (Skalse et al., 2022) leading to different behavior in environments and in the wild becomes a crucial concern.

4.3 Fine-tune everything?

While some hypothesize that persistent context gaps can be mitigated by fine-tuning models for each workload, user, or deployment setting (e.g., via LoRA (Hu et al., 2022)). However, several unresolved challenges persist: first, continual or repeated fine-tuning risks catastrophic forgetting of base capabilities (Luo et al., 2025a)—mitigating this requires careful regularization, curriculum design, and learning-rate scheduling (Ibrahim et al., 2024), which might yield brittle or unstable behavior under distribution shifts. Second, even if such adaptation were economically viable, fragmenting knowledge into per-application silos limits an agent's ability to orchestrate tasks *across* them. Third, parameter updates reduce deployment agility (Agrawal et al., 2025): models cannot easily adapt to immediate feedback, nor can they selectively forget outdated or unhelpful information without retraining. Finally, as base models continue to improve in general reasoning and execution, the marginal gains from additional fine-tuning on narrow contexts become increasingly unclear, particularly when failures stem from missing or implicit context rather than insufficient model capacity.

4.4 Just encode context into prompts or memories

It might easy to confuse the current position as "RAG": relevant context e.g., tool documentation, workflows, user preferences, and prior interactions, can simply be injected into prompts or stored in external memories, eliminating the need for new learning mechanisms. While effective in narrow settings, this approach faces scalability and robustness challenges, as discussed in §3. We consider prompt- and memory-based approaches as valuable components, but in and of itself, they are insufficient. Effective personal agents require mechanisms that can acquire, validate, and apply context dynamically during deployment, without relying solely on brittle prompt growth or repeated retraining.

5 Conclusion and Call to Action

We urge the community to reorient research away from exclusively scaling training, which helps some applications, and move toward enabling *deployment-time context acquisition* which can help make personal agents practical. Our findings suggest that many persistent failures do not stem from insufficient reasoning capacity, but from missing, implicit, or rapidly evolving context that cannot be reliably encoded during training. Addressing this gap requires new system designs, benchmarks, and evaluation protocols that reward agents for identifying uncertainty, involving humans or external sources appropriately, and internalizing acquired knowledge without updating model parameters.

Concretely, we call for: (i) benchmarks that surface context gaps and measure an agent's ability to recover from them during deployment; (ii) mechanisms for confidence estimation and external involvement that go beyond heuristic prompt engineering; and (iii) learning techniques that allow generalizing deployment-acquired context across tasks without brittle fine-tuning. Progress on personal agents will depend less on training ever-larger models, and more on

end-to-end systems that can identify what they do not know and acquire that knowledge once deployed.

References

Bespoke labs. https://www.bespokelabs.ai/.

Agrawal, L. A., Tan, S., Soylu, D., Ziems, N., Khare, R., Opsahl-Ong, K., Singhvi, A., Shandilya, H., Ryan, M. J., Jiang, M., et al. Gepa: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*, 2025.

Ai, Q., Dumais, S. T., Craswell, N., and Liebling, D. Characterizing email search using large-scale behavioral logs and surveys. In *Proceedings of the 26th International Conference on World Wide Web*, pp. 1511–1520, 2017.

Aljalbout, E., Xing, J., Romero, A., Akinola, I., Garrett, C. R., Heiden, E., Gupta, A., Hermans, T., Narang, Y., Fox, D., et al. The reality gap in robotics: Challenges, solutions, and best practices. *Annual Review of Control, Robotics, and Autonomous Systems*, 9, 2025.

Anthropic. Model context protocol. https://modelcontextprotocol.io, a.

Anthropic. Claude code: AI coding agent for terminal and IDE. https://www.anthropic.com/claude-code, b.

Anysphere. Cursor: The AI code editor. https://cursor.com.

Bai, G., Liu, J., Bu, X., He, Y., Liu, J., Zhou, Z., Lin, Z., Su, W., Ge, T., Zheng, B., et al. Mt-bench-101: A fine-grained benchmark for evaluating large language models in multi-turn dialogues. *arXiv preprint arXiv:2402.14762*, 2024.

Bandi, C., Hertzberg, B., Boo, G., Polakam, T., Da, J., Hassaan, S., Sharma, M., Park, A., Hernandez, E., Rambado, D., et al. Mcp-atlas: A large-scale benchmark for tool-use competency with real mcp servers. 2025.

Bansal, G., Wu, T., Zhou, J., Fok, R., Nushi, B., Kamar, E., Ribeiro, M. T., and Weld, D. Does the whole exceed its parts? the effect of ai explanations on complementary team performance. In *Proceedings of the 2021 CHI conference on human factors in computing systems*, pp. 1–16, 2021.

Barres, V., Dong, H., Ray, S., Si, X., and Narasimhan, K. τ^2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*, 2025.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., C. Martin, R., Mellor, S., Schwaber, K., Sutherland, J., and Thomas, D. Manifesto for agile software development. https://agilemanifesto.org/.

Cemri, M., Pan, M. Z., Yang, S., Agrawal, L. A., Chopra, B., Tiwari, R., Keutzer, K., Parameswaran, A., Klein, D., Ramchandran, K., et al. Why do multi-agent llm systems fail? *arXiv preprint arXiv:2503.13657*, 2025.

Draucker, C. B., Martsolf, D. S., Ross, R., and Rusk, T. B. Theoretical sampling and category development in grounded theory. *Qualitative health research*, 17(8): 1137–1148, 2007.

Etzioni, O. and Weld, D. A softbot-based interface to the internet. *Communications of the ACM*, 37(7):72–76, 1994.

Falahati, A., Amiri, M. M., Larson, K., and Golab, L. The alignment game: A theory of long-horizon alignment through recursive curation. *arXiv preprint arXiv:2511.12804*, 2025.

Gan, T. and Sun, Q. Rag-mcp: Mitigating prompt bloat in llm tool selection via retrieval-augmented generation. *arXiv preprint arXiv:2505.03275*, 2025.

Ge, T., Chan, X., Wang, X., Yu, D., Mi, H., and Yu, D. Scaling synthetic data creation with 1,000,000,000 personas. *arXiv preprint arXiv:2406.20094*, 2024.

Glaser, B. and Strauss, A. *Discovery of grounded theory: Strategies for qualitative research*. Routledge, 2017.

Google DeepMind. Gemini deep research. `https://gemini.google/overview/deep-research/`.

Gunjal, A., Wang, A., Lau, E., Nath, V., He, Y., Liu, B., and Hendryx, S. Rubrics as rewards: Reinforcement learning beyond verifiable domains. *arXiv preprint arXiv:2507.17746*, 2025.

Guo, Z., Cheng, S., Wang, H., Liang, S., Qin, Y., Li, P., Liu, Z., Sun, M., and Liu, Y. Stabletoolbench: Towards stable large-scale benchmarking on tool learning of large language models. *arXiv preprint arXiv:2403.07714*, 2024.

He, Y., Li, W., Zhang, H., Li, S., Mandyam, K., Khosla, S., Xiong, Y., Wang, N., Peng, X., Li, B., et al. Advancedif: Rubric-based benchmarking and reinforcement learning for advancing llm instruction following. *arXiv preprint arXiv:2511.10507*, 2025.

Horvitz, E. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pp. 159–166, 1999.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Ibrahim, A., Thérien, B., Gupta, K., Richter, M. L., Anthony, Q., Lesort, T., Belilovsky, E., and Rish, I. Simple and scalable strategies to continually pre-train large language models. *arXiv preprint arXiv:2403.08763*, 2024.

Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., and Narasimhan, K. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

Kadavath, S., Conerly, T., Askell, A., Henighan, T., Drain, D., Perez, E., Schiefer, N., Hatfield-Dodds, Z., DasSarma, N., Tran-Johnson, E., et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Kate, K., Pedapati, T., Basu, K., Rizk, Y., Chenthamarakshan, V., Chaudhury, S., Agarwal, M., and Abdelaziz, I. Longfunceval: Measuring the effectiveness of long context models for function calling. *arXiv preprint arXiv:2505.10570*, 2025.

Khandkar, S. H. Open coding. *University of Calgary*, 23 (2009):2009, 2009.

Kim, J. Y., Craswell, N., Dumais, S., Radlinski, F., and Liu, F. Understanding and modeling success in email search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 265–274, 2017.

Kuhn, L., Gal, Y., and Farquhar, S. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. *arXiv preprint arXiv:2302.09664*, 2023.

Li, M., Zhao, Y., Yu, B., Song, F., Li, H., Yu, H., Li, Z., Huang, F., and Li, Y. Api-bank: A comprehensive benchmark for tool-augmented llms. *arXiv preprint arXiv:2304.08244*, 2023.

Lin, S., Hilton, J., and Evans, O. Teaching models to express their uncertainty in words, 2022. URL `https://arxiv.org/abs/2205.14334`.

Lin, Y., Lin, H., Xiong, W., Diao, S., Liu, J., Zhang, J., Pan, R., Wang, H., Hu, W., Zhang, H., et al. Mitigating the alignment tax of rlhf. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 580–606, 2024.

Luo, Y., Yang, Z., Meng, F., Li, Y., Zhou, J., and Zhang, Y. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *IEEE Transactions on Audio, Speech and Language Processing*, 2025a.

Luo, Z., Shen, Z., Yang, W., Zhao, Z., Jwalapuram, P., Saha, A., Sahoo, D., Savarese, S., Xiong, C., and Li, J. Mcp-universe: Benchmarking large language models with real-world model context protocol servers. *arXiv preprint arXiv:2508.14704*, 2025b.

Mozannar, H. and Sontag, D. Consistent estimators for learning to defer to an expert. In *International conference on machine learning*, pp. 7076–7087. PMLR, 2020.

NANDA, M. State of ai in business 2025. 2025.

Pan, M. Z., Arabzadeh, N., Cogo, R., Zhu, Y., Xiong, A., Agrawal, L. A., Mao, H., Shen, E., Pallerla, S., Patel, L., et al. Measuring agents in production. *arXiv preprint arXiv:2512.04123*, 2025.

Paramanayakam, V., Karatzas, A., Anagnostopoulos, I., and Stamoulis, D. Less is more: Optimizing function calling for llm execution on edge devices. In *2025 Design, Automation & Test in Europe Conference (DATE)*, pp. 1–7. IEEE, 2025.

Patil, S. G., Mao, H., Yan, F., Ji, C. C.-J., Suresh, V., Stoica, I., and Gonzalez, J. E. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*.

Patil, S. G., Zhang, T., Wang, X., and Gonzalez, J. E. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37: 126544–126565, 2024.

Prince, M. Does active learning work? a review of the research. *Journal of engineering education*, 93(3):223–231, 2004.

Ross, S., Gordon, G., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Shao, R., Asai, A., Shen, S. Z., Ivison, H., Kishore, V., Zhuo, J., Zhao, X., Park, M., Finlayson, S. G., Sontag, D., et al. Dr tulu: Reinforcement learning with evolving rubrics for deep research. *arXiv preprint arXiv:2511.19399*, 2025.

Skalse, J., Howe, N., Krasheninnikov, D., and Krueger, D. Defining and characterizing reward gaming. *Advances in Neural Information Processing Systems*, 35:9460–9471, 2022.

Song, D., Zhou, Z., Wang, Z., Huang, Y., Chen, S., Kou, B., Ma, L., and Zhang, T. An empirical study of code generation errors made by large language models. In *7th Annual Symposium on Machine Programming*, 2023.

Suzgun, M., Yuksekgonul, M., Bianchi, F., Jurafsky, D., and Zou, J. Dynamic cheatsheet: Test-time learning with adaptive memory. *arXiv preprint arXiv:2504.07952*, 2025.

Trivedi, H., Khot, T., Hartmann, M., Manku, R., Dong, V., Li, E., Gupta, S., Sabharwal, A., and Balasubramanian, N. Appworld: A controllable world of apps and people for benchmarking interactive coding agents. *arXiv preprint arXiv:2407.18901*, 2024.

Winston, C. and Just, R. A taxonomy of failures in tool-augmented llms. In *2025 IEEE/ACM International Conference on Automation of Software Test (AST)*, pp. 125–135. IEEE, 2025.

Wu, F. and Choi, Y. The invisible leash: Why rlvr may not escape its origin. In *2nd AI for Math Workshop@ ICML 2025*, 2025.

Wu, S., Zhao, S., Huang, Q., Huang, K., Yasunaga, M., Cao, K., Ioannidis, V. N., Subbian, K., Leskovec, J., and Zou, J. Avatar: Optimizing llm agents for tool usage via contrastive reasoning. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 25981–26010. Curran Associates, Inc., 2024. doi: 10.52202/079017-0817.

Xie, A., Lee, L., Xiao, T., and Finn, C. Decomposing the generalization gap in imitation learning for visual robotic manipulation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3153–3160. IEEE, 2024.

Xiong, M., Hu, Z., Lu, X., Li, Y., Fu, J., He, J., and Hooi, B. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*, 2024. URL https://openreview.net/forum?id=gjeQKFxFpZ.

Yang, J., Jimenez, C. E., Wettig, A., Lieret, K., Yao, S., Narasimhan, K., and Press, O. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

Yao, S., Shinn, N., Razavi, P., and Narasimhan, K. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Yu, T., Ji, B., Wang, S., Yao, S., Wang, Z., Cui, G., Yuan, L., Ding, N., Yao, Y., Liu, Z., et al. Rlpr: Extrapolating rlvr to general domains without verifiers. *arXiv preprint arXiv:2506.18254*, 2025.

Yue, Y., Chen, Z., Lu, R., Zhao, A., Wang, Z., Song, S., and Huang, G. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025.

Zhang, A. L., Kraska, T., and Khattab, O. Recursive language models. *arXiv preprint arXiv:2512.24601*, 2025a.

Zhang, Q., Hu, C., Upasani, S., Ma, B., Hong, F., Kamanuru, V., Rainton, J., Wu, C., Ji, M., Li, H., et al. Agentic context engineering: Evolving contexts for self-improving language models. *arXiv preprint arXiv:2510.04618*, 2025b.

Zhang, S., Yin, M., Zhang, J., Liu, J., Han, Z., Zhang, J., Li, B., Wang, C., Wang, H., Chen, Y., et al. Which agent causes task failures and when? on automated failure attribution of llm multi-agent systems. *arXiv preprint arXiv:2505.00212*, 2025c.

Zhao, R., Meterez, A., Kakade, S., Pehlevan, C., Jelassi, S., and Malach, E. Echo chamber: Rl post-training amplifies behaviors learned in pretraining. *arXiv preprint arXiv:2504.07912*, 2025.

A AFT: Definitions

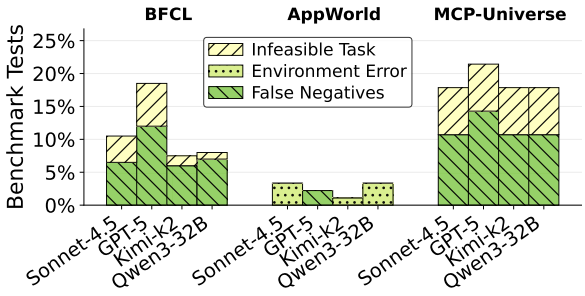

Figure 12: Prevalance of different types of benchmark errors in our final dataset

Figure 13 catalogs all failure types uncovered in **AFT**.

B Benchmark Errors

Figure 12 shows the prevalence of the different types of benchmark errors in our dataset.

B.1 False Negatives

We identified several distinct reasons for false-negative evaluations:

- *Trajectory Matching:* BFCL's validator compared the agent's function call trajectory against a gold subset, in addition to matching the final state after execution, and although crucial to ensure that the required functions are actually executed and agents do not hallucinate or memorize the responses, frequently resulted in false negatives, when task execution deviated slightly: e.g., rename-then-move a file, instead of move-then-rename.
- *Benign Formatting Differences:* AppWorld and MCP-Universe used simple text matching instead of specialized parsers to validate model-generated CSV/code files and flagged benign differences, such as if the file ends with a newline or not, unexpected wrapping of values inside quotes ("1", "2").
- *Validator Code Bugs:* MCP-Universe attempted calling MCP tools directly to validate agent affected changes, but failed due to updates to GitHub MCP server's exposed schema. We changed the validator to leverage GitHub REST APIs, that provide stable APIs and are more appropriate for application code, whereas MCP tools are usually orchestrated dynamically by an LLM and can be updated frequently.

B.2 Infeasible Tasks

Apart from some obvious bugs e.g., not sufficiently checking pre-conditions e.g., ensuring sufficient balance before

Category	Definition
Agent Errors	Test failed because agent's output did not satisfy benchmark's reasonable expectations.
Unsuccessful Completion	Agent reasonably completed the task instructions but did not meet requirements exactly.
Ambiguous Instructions	Agent's interpretation of task was reasonable but differed from benchmark's expectation.
Formatting Error	Only the agent's final output format differed from benchmark's expectation.
Output State Mismatch	Agent completed task reasonably but the final state did not match requirements exactly.
Incomplete Execution	Agent failed to complete the provided task instructions.
Stopped For User Input	Agent unreasonably paused to request clarification instead of proceeding autonomously.
Resp. / Param. Fabrication	Agent *hallucinated* response or parameter values not grounded in the current context.
Function Invocation Error	Agent was unable to execute function calls correctly: syntax or semantic errors.
Incorrect Function Selection	Agent selected wrong functions or wrong sequence; includes sub-categories below
Missing Tacit Knowledge	Agent lacked basic knowledge required for completing task not stated in instructions or prompts.
Skipped Sub-Task	Agent omitted necessary intermediate steps inferable from task description
Function Docs Ignored	Agent did not select correct function despite explicit documentation included regarding usage.
Pagination Error	Agent failed to retrieve all pages of paginated API responses
Benchmark Errors	Test failed due to benchmark setup / evaluation issues rather than agent failures.
False Negatives	Agent completed task correctly but benchmark incorrectly flagged as failure
Infeasible Task	Task impossible to complete given initial state; agent correctly identified infeasibility
Environment Error	Mock environment behavior diverged from realistic expectations, preventing valid solutions
System / API Errors	API request failures from exceeding provider limits on functions per turn or API rate limits

Figure 13: AFT taxonomy categories and definitions; indentation indicates nesting of hierarchical categories. Each failure log must belong to exactly one of the top level categories, e.g,. unsuccessful completion, incomplete execution, benchmark errors, etc., but can belong to multiple sub-categories.

executing a transaction [5], we found test-cases, feasible initially, had turned infeasible due to temporal effects; for instance, some MCP-Universe tests required forking a particular repository which no longer exists, and BFCL instructions (which use mock fixtures) had hardcoded dates from 2024 for flight-booking and cancellation tasks; more recent models–which either have a later knowledge cutoff or access to the current timestamp from the system prompt (e.g., GPT-5)—refused to perform actions in the past. For our experiments, we updated all the dates in such task instructions and ground truth evaluations to ensure they were in the future. We preferred this over AppWorld's approach of providing the current timestamp in the system prompt and instructing the models to ignore all other timing information, since that gives away the evaluative nature of the task, and although the impact is not fully understood, such awareness might lead to models performing differently than for real-world user queries. [6]

B.3 Environment Errors

- BFCL functions did not return meaningful error messages in some cases when models performed invalidation actions, which limited the models' ability to self-correct based on environmental feedback.
- AppWorld's environment limited the # of execution turns per test-case to 40 when the # function invocations required to complete the tasks was significantly higher Further, their system prompts *encouraged* models to aggressively batch dispatch multiple function calls within a single, which significantly increased the failure rates, as present models capabilities to accurately dispatch parallel function calls can be significantly worse than executing them sequentially; we relaxed these constraints and removed the corresponding instruction from the prompt allowing the models to switch between parallel and sequential execution naturally based on their capabilities, which improved the accuracy of GPT-5 and Claude on these tasks significantly.
- AppWorld APIs enabled combining arguments which led to invalid result-ordering e.g., keyword filtering, pagination, and temporal ordering generated result pages split based on keyword similarity but ordered by time. We found these invalid cases concentrated within a few models, and the AppWorld implementation has since removed the problematic keyword similarity argument from these APIs.
- AppWorld APIs split large response arrays across multiple pages, but did not provide any metadata (e.g., a next cursor), which is standard practice in pagination implementations, to inform the caller about the presence of additional data. Instead, models were expected to check if # returned entries matches the page size, or always check the next pages to check if there are additional results. While their system prompt instructed the models of such expectations, which were sufficient for GPT-5 and Sonnet 4.5, other models frequently made pagination errors, which we report separately
- AppWorld's mock clocks also run significantly faster than wall-time, which forced repeated logins due to token ex-

[5] We worked with the BFCL maintainers to fix such test-cases; our presented results include these fixes.

[6] We did not change AppWorld's test-cases due to their enviroment and validator implementations.

piry; while models should be be able to handle this, we highlight this as an another example where production environments and mock environments behave different.