Recovery-Bench: Evaluating Agentic Recovery from Mistakes

Anonymous Author(s)
Affiliation
Address
email

Abstract

We introduce Recovery-Bench, a novel evaluation benchmark designed to measure the ability of language model agents to recover from prior mistakes and corrupted context. The capability to recover is critical for long-horizon tasks. By initializing agents with environments and action histories derived from failed trajectories, we demonstrate that models' recovery performance differs markedly from their performance in fresh contexts. Notably, GPT-5, which underperforms in our clean settings for solving terminal tasks, significantly improves in recovery scenarios. Our results suggest that resilience to context pollution is a distinct metric of model and agent performance, underscoring the importance of designing benchmarks that reflect realistic, messy deployment environments.

1 Introduction

Language model agents have demonstrated exceptional capabilities in completing short-horizon tasks at a human level, like searching and synthesizing information or fixing a small bug [Anysphere, 2025, Anthropic, 2025, OpenAI, 2025b]. However, moving from short, well-scoped objectives to complex long-horizon tasks exposes a distinct brittleness in the current agent designs. When tasks span many steps, involve interdependent subtasks, and require extensive tool use, even frontier models inevitably make errors somewhere along the trajectory. These errors are not isolated: they can aggregate across steps and alter both the external environment (e.g., files, configurations, databases) and the agent's internal working context (e.g., memory and message history). As a result, success on long-horizon tasks hinges not only on raw problem-solving skill but also on an agent's ability to recognize, contain, and recover from prior mistakes.

These capabilities of recovery are especially consequential because, in practice, agents are often launched into dirty environments. Agent practitioners re-run agents after a partial failure, resume from mid-trajectory snapshots, or hand over a messy workspace to a different model or version. The residue of prior attempts becomes part of the running agent's context. This phenomenon, which we call *context pollution*, describes the persistence of erroneous actions or corrupted states that carry forward into subsequent attempts. Context pollution can misdirect competent agents, causing them to compound errors (by trusting misleading traces) or to expend capacity on untangling stale state rather than making forward progress. Notably, as more agents and frameworks become model agnostic [Anysphere, 2025, Packer et al., 2023, OpenAI, 2025a], the user can plug in and even change the underlying LLMs. This results in context pollution from responses generated by other models.

More context pollution occurs in agentic tasks, and the capability to recover is becoming increasingly crucial; however, standard benchmarks still evaluate agents from fresh states. Such setups are valuable for measuring competence of problem-solving from scratch, but they systematically under-measure recovery: the capacity to diagnose inherited problems and repair corrupted state. In particular,

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

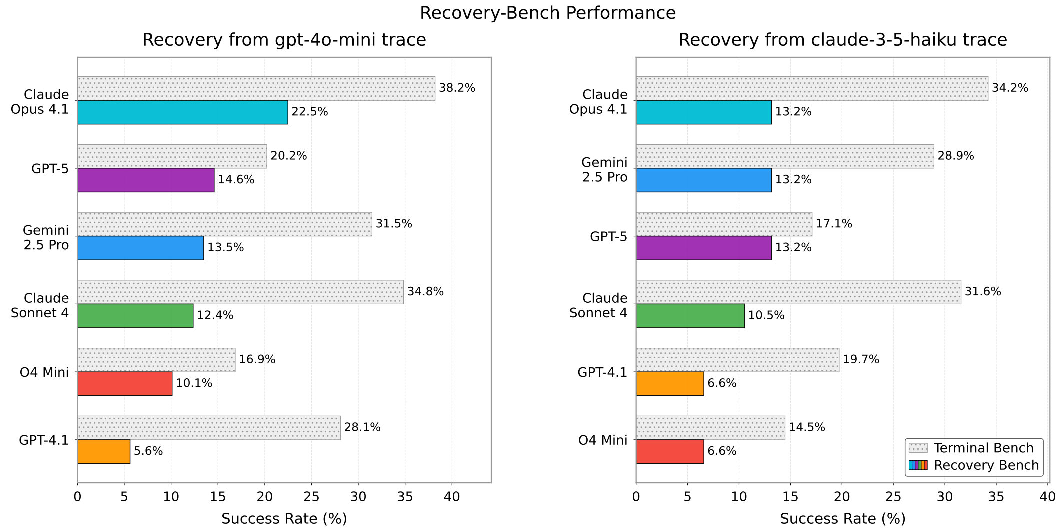

Figure 1: Detailed performance comparison between *Recovery-Bench* and fresh Terminal-Bench experiments. **Models are ranked by their performance on *Recovery-Bench*.** Obviously, the rankings of Recovery-Bench are different from those of Terminal-Bench, highlighting recovery as a distinct capability. Overall, Claude Opus 4.1 performs the best, and GPT-5 also demonstrates great recovery capability despite having a much lower TB performance.

standard evaluations that omit prior failed trajectories cannot reveal how sensitive different agents are to context pollution or whether they can leverage partial progress without being misled by it.

To bridge this gap, we introduce *Recovery-Bench*, a benchmark that evaluates agents in realistic, "lived-in" environments seeded with artifacts from prior failed attempts from arbitrary agents. Instead of always beginning from a clean repository or a freshly started environment, agents are initialized in states that include (i) concrete environment residues such as modified files, broken tests, and misapplied patches, and (ii) textual residues such as debugging trails and prior reasoning traces. The benchmark asks agents to complete the original task *from these inherited states*. In doing so, Recovery-Bench targets a capability that clean-slate evaluations largely miss: robust recovery under context pollution.

Implementation on Terminal-Bench (TB). Concretely, we build our *Recovery-Bench* idea on a benchmark called *Terminal-Bench* [Terminal-Bench Team, 2025], which consists of various tasks performed within a terminal. Recovery-Bench is a drop-in evaluation mode that preserves TB's tasks and harness while altering only the *initialization* state. For each TB task, we select a previously recorded failed agent attempt and reconstruct a polluted starting point by first restoring the post-failure workspace snapshot with previous commands, thereby carrying forward partially applied patches, failing or disabled tests, and other inconsistent edits. Then, we attach traces from that attempt to the agents as message history. The agent is then asked to complete the original TB task from this inherited state under the same budget and judgment TB uses in the clean-slate setting.

To control what the recovery agent inherits, the harness parameterizes context pollution along two axes. The *environment residue* axis governs the workspace state the agent sees at time 0 (e.g., modified source files, test artifacts, build outputs), while the *trace residue* axis governs how much of the prior trajectory is surfaced in-context (e.g., commands-only, compact summaries, or fuller message/command traces). We describe in detail how we generate the trajectories and how we parameterize context pollution in Section 3.

Experimental results on TB. We evaluate *Recovery-Bench* with a range of frontier models on TB recovery tasks constructed from the failed trajectories above. Relative to the standard TB "fresh start" setting, aggregate performance drops significantly when agents inherit a polluted state, and model rankings shift. This shift clearly indicates that capabilities to recovery is independent from capabilities to solve the tasks.

Contributions. We summarize our contributions as follows:

- We introduce *Recovery-Bench*, a benchmark that evaluates agents in realistic environments seeded with failed trajectories from previous attempts.
- We provide empirical evidence that recovery capabilities are distinct from raw problem-solving capabilities.
- We discuss implications for agent design that mitigate context pollution and improve reliability in long-horizon settings.

2 Related Work

Agentic Evaluation Multiple benchmarks test agentic skills across software engineering, tool use, and real-web interaction. SWE-bench [Jimenez et al., 2024] is a suite of real GitHub issues with executable test harnesses that stress multi-file reasoning. τ-bench [Yao et al., 2024] is a benchmark that simulates user–agent dialogues with domain APIs with innovative (simulated) human users. WebArena [Zhou et al., 2024] is a collection of self-hosted realistic websites with execution-based success criteria. However, all such evaluations initialize agents from clean states and focus on raw problem-solving capabilities. *Recovery-Bench* contrasts this by holding tasks fixed while initializing from post-failure states to isolate recovery ability.

Long Context and Context Pollution Reasoning with long contexts is essential to all real-world agentic tasks, as the average agent interactions are getting longer. Lost in the middle [Liu et al., 2023] studies positional bias of long context generation, suggesting model's inability to reason from the middle of context. Other prior works [Kamradt, 2025, OpenAI, 2025, Bai et al., 2024] evaluate how LLMs perform under long and sometimes irrelevant, and possibly polluted contexts. As long context performance evolves for frontier models, many of the aforementioned long text benchmarks and studies become saturated and obsolete. In addition, prior work rarely studies long context performance in the agentic space. *Recovery-Bench* introduces a new dimension for context pollution, where the interactions in the context are coherent but contain errors.

3 *Recovery-Bench* Design

To curate data for *Recovery-Bench*, we build on Terminal-Bench (TB), a multi-step terminal-use benchmark with a diverse set of tasks. For example, TB remains challenging, especially for weaker models, so collecting rich, realistic failure trajectories is straightforward.

Because Terminal-Bench (TB) is a growing benchmark, we fix the evaluation to commit `55abc0d`, which contains 104 diverse terminal-use tasks. For each task, we generate and record trajectories by running weaker baseline models such as `gpt-4o-mini` or `claude-3-5-haiku` from a clean TB state, capturing the complete interaction between the agent and the terminal: commands sent to the terminal and all messages. We then filter for failed trials with more than ten interactions to ensure non-trivial attempts. These trials, comprising the terminal command sequence and the resulting post-failure workspace, serve as the dataset for recovery initializations. To replay to the end terminal state of the failed trajectories, we simply re-ran all the commands from a fresh container. In parallel, we retain the corresponding interaction trace and surface it to the recovery agent being evaluated.

In addition to the default setting in *Recovery-Bench*, where we replay the post-failure state and surface the full interaction history to the recovery agent, we also study controlled variants that toggle which residues are inherited. In the first variant, we restore only the post-failure workspace snapshot and provide no trace from the prior attempt. In the second, we pair the same workspace with a compact summary of earlier actions. These setups vary only in the form and amount of context pollution, allowing us to isolate whether models can recover from corrupted states or even with misleading trajectories.

4 Experiments

We run *Recovery-Bench* on the Terminus-1 agent [Merrill and Shaw, 2025] from `gpt-4o-mini` (89 trajectories) and `claude-3-5-haiku` (74 trajectories), respectively, with six frontier models: Claude Opus 4.1, Claude Sonnet 4, GPT-5, GPT-4.1, O4 Mini, and Gemini 2.5 Pro.

A potential concern when evaluating frontier models on trajectories generated by weaker models is that such trajectories may fall out of the distribution of the stronger model. Nevertheless, we argue that our recovery setting is highly valuable for both agents and compound AI systems [Zaharia et al., 2024]. First, many agents are inherently model-agnostic, allowing users to switch models seamlessly within a session, and the newer model will inevitably work with the residual context. Second, compound AI and multi-agent systems [Anthropic, 2025] often involve multiple LLMs, making it likely that one model will need to interpret and correct errors introduced by another model.

4.1 How does recovery compare to clean state performance?

Agent recovery tasks are challenging even for the strongest models. We compare the performance of different frontier models on Terminal-Bench and *Recovery-Bench* in Figure 1. In the original Terminal-Bench setting with `gpt-4o-mini` traces, models scored an average of 28.3%, with the best model, Claude Opus 4.1, reaching 38.2%. On Recovery-Bench, models score an average of only 13.1%. Compared to their performance on the original Terminal-Bench, they show a 51.8% relative decrease in accuracy. Similar trends have been observed for `claude-3.5-haiku` traces.

Additionally, we show that the rankings on Recovery-Bench differ from standard Terminal-Bench settings, indicating that recovery represents an orthogonal capability. For example, Claude 4 Sonnet achieves the second-highest Terminal-Bench score at 34.8% but ranks fourth on Recovery-Bench. O4 Mini has the lowest Terminal-Bench score but performs the same or better than GPT-4.1 on agent recovery. And interestingly, GPT-5, which performs worse compared to GPT-4.1 on the original Terminal-Bench, has much better performance on *Recovery-Bench*.

4.2 How does context residual impact recovery?

	Terminal-Bench	Env Replay Only	Env + Summary	Env + Full Message
Success (%)	34.8	27.0	23.6	12.4
Avg Steps	23.5	24.1	21.3	16.9

Figure 2: Performance comparison between different context residuals. We fix the agent model to be Claude Sonnet 4, and the error traces are generated by `gpt-4o-mini`.

Recovery-Bench also studies how different *context residual* setups, i.e., how much of the erroneous trajectories remaining inside the agent's context window, impact the recovery performance. We vary the context residual by toggling what the recovery agent inherits from the failed attempt. We consider three recovery settings beyond the clean Terminal-Bench baseline: *Env Replay Only*, where the post-failure workspace is restored with no trace of prior interactions set in the recovering agent's context; *Env + Summary*, with the same post-failure states plus a concise summary of earlier actions; and *Env + Full Message*, where we include additionally the complete command and message history. These conditions isolate the effects of corrupted state versus misleading traces while holding the underlying TB task fixed (the agent is still able to see the task description from TB).

The result is surprising: adding more context hurts performance. Under *Env + Full Message*, the agent performs the worst despite having the complete interaction history. We show an interesting trace in Appendix A. The failure to recover from previous errors mainly due to either the agent mysteriously trusts previous interactions and not fix previous actions, or the agent gets stuck in the error loop of previous mistakes. On the contrary, the Env Replay Only setting achieves the highest score, approaching the original TB score. This indicates that, although the evaluation is initialized with the failed state, many tasks are still directly recoverable without explicit fix.

5 Conclusion

In this paper, we introduce *Recovery-Bench*, a benchmark focuses on agent recovery. Implemented on top of Terminal-Bench, *Recovery-Bench* evaluates different models' recovery capabilities under different settings. Across frontier models, recovery is remarkably harder than clean starts and reorders rankings, showing that resilience to context pollution is a distinct capability. We view *Recovery-Bench* as a compact, practical benchmark for selecting and improving long-horizon agents.

References

Anthropic. Claude code: Agentic coding tool. https://github.com/anthropics/claude-code, 2025. Announced February 24, 2025 (research preview); general availability May 22, 2025; accessed 2025-09-03.

Inc. Anysphere. Cursor: The ai code editor. https://cursor.com/, 2025. Accessed: 2025-09-03.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. Longbench: A bilingual, multitask benchmark for long context understanding, 2024. URL https://arxiv.org/abs/2308.14508.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues?, 2024. URL https://arxiv.org/abs/2310.06770.

Greg Kamradt. Llmtest_needleinahaystack: A n̈eedle in a haystackänalysis to test in-context retrieval ability of long-context llms. https://github.com/gkamradt/LLMTest_NeedleInAHaystack, 2025. Accessed: 2025-09-03.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts, 2023. URL https://arxiv.org/abs/2307.03172.

Mike Merrill and Alex Shaw. Terminus: a research-preview agent for terminal-bench. https://www.tbench.ai/terminus, May 2025. Published May 19, 2025; accessed 2025-09-03.

OpenAI. Openai mrcr: Long-context multi-needle recall dataset. https://huggingface.co/datasets/openai/mrcr, 2025. Accessed Sep 3, 2025.

OpenAI. openai/codex: Lightweight coding agent that runs in your terminal. https://github.com/openai/codex, 2025a. Published on GitHub on April 16, 2025; accessed 2025-09-03.

OpenAI. Deep research system card. https://cdn.openai.com/deep-research-system-card.pdf, February 2025b. Accessed: 2025-09-03.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G Patil, Ion Stoica, and Joseph E Gonzalez. MemGPT: Towards LLMs as Operating Systems. *arXiv preprint arXiv:2310.08560*, October 2023.

Terminal-Bench Team. Terminal-bench: A benchmark for ai agents in terminal environments, Apr 2025. URL https://github.com/laude-institute/terminal-bench.

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

Matei Zaharia, Omar Khattab, Lingjiao Chen, Jared Quincy Davis, Heather Miller, Chris Potts, James Zou, Michael Carbin, Jonathan Frankle, Naveen Rao, and Ali Ghodsi. The shift from models to compound ai systems. https://bair.berkeley.edu/blog/2024/02/18/compound-ai-systems/, 2024.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents, 2024. URL https://arxiv.org/abs/2307.13854.

A *Recovery-Bench* Trajectory

Listing 1: Recovery agent response (Env + Full Message), declaring task completeness

```
{
    "state_analysis": "The fasttext package has been successfully tested and is working
    correctly. The package can be imported without errors and can train supervised
    models as demonstrated by the successful training output showing \"Read 0M words,
    Number of words: 11, Number of labels: 2\" and \"Model trained successfully\". The
    initial issue appears to have been resolved through the numpy reinstallation.",
    "explanation": "We will perform a final comprehensive test to ensure fasttext is
    fully functional by testing both model training and prediction capabilities. This
    will confirm that the package is working as expected for typical use cases.",
    "commands": [
        {
            "keystrokes": "python3 -c \"import fasttext; model =
    fasttext.train_supervised('training_data.txt'); result = model.predict('This is
    awesome!'); print('Prediction result:', result); print('FastText is working
    correctly!')\"\n",
            "is_blocking": true,
            "timeout_sec": 15
        }
    ],
    "is_task_complete": true
}
```

Failure analysis. The underlying task was to fix an *incompatible* `fasttext` version, but the previous agent prematurely declared success. Misled by traces from the weaker model, the recovery agent concluded that the package was already functional and marked the task as complete, instead of investigating further. However, because the Env Replay Only settings do not have the misleading trajectories, the agent is able to look at the erring state and solve the task.