

# Causal Decomposition of LLM Agent Failure via Oracle-State Intervention

Leqi Jia<sup>1</sup>, Muhammad Saadat<sup>2</sup>, Marlin Wong<sup>3</sup>, Adithya Venkatesh<sup>4</sup>, Kiran Njjer<sup>5</sup>

The Harker School<sup>1</sup> Canton High School<sup>2</sup> University of Washington<sup>3</sup>

BASIS Independent Fremont<sup>4</sup> Stanford School of Medicine<sup>5</sup>

Correspondence: [28rockyj@students.harker.org](mailto:28rockyj@students.harker.org)

## Abstract

LLM agents degrade as trajectories lengthen, motivating current research on context management methods that compress, filter, or restructure history. All of these methods share the untested assumption that context degradation is the primary bottleneck of agentic systems. To test this hypothesis, we introduce an oracle-state intervention framework that decomposes agent failure modes, isolates context as a causal variable by replacing a failing agent’s accumulated history with ground-truth structured state, and measures recovery. Across 420 failed WebArena trajectories with two models (Qwen3.5-72B and GPT-5.2), we find that context management is effective but not in the way commonly assumed in recent literature. In the default WebArena Qwen and GPT configurations, replacing context after failure recovers fewer than 9% of failed trajectories, while 58%-74% of failures remain unrecovered after full oracle intervention; human annotations independently assign most failures to capability-caused labels. Yet continuous oracle state throughout the trajectory improves success by 19-31 percentage points, and the gap between oracle-maintained and self-generated state widens from 7% to 18% as trajectories lengthen. Context management, hence, works preventively as it slows the accumulation of errors along a trajectory, but rarely rescues an agent that has already gone off track. Our findings suggest that progress on agentic systems will require improving planning and execution capabilities alongside context engineering, rather than treating context quality as the dominant bottleneck.

## 1 Introduction

Agent accuracies fall below 50% within the first 15 steps as models spiral into their earlier errors (Sinha et al., 2025), and even state-of-the-art models achieve only about 12% success on desktop tasks, whereas humans achieve 72% (Xie et al., 2024). When an agent diverges from its optimal

trajectory or abandons a partially completed task, the trajectory alone does not inform us whether the failure is attributed to a corrupted context or to a limit of the agent’s ability to act correctly given any context at all. This distinction determines what kind of intervention can help prevent failures—if context is the bottleneck, the field’s emphasis on context engineering is well-motivated; however, if not, no amount of context optimization will fully close the gap.

We conduct our experiments on WebArena (Zhou et al., 2024), an environment where we have access to ground-truth states. We build oracle states in WebArena by structuring information related to the goal, completed or unfinished tasks, true environment state, constraints, failures, and key extracted facts. The oracle generator, Claude Sonnet 4.6, has unrestricted access to the live environment, which allows it to identify which observations in the agent’s accumulated context are outdated and which attributions in its reasoning are incorrect. Experiment 1 tests whether improved context can help a failing agent recover. At the point of failure, we either replace the agent’s context with the oracle state or preserve the original context while resetting the environment. Experiment 2 examines whether improved context can prevent failure in the first place. We run trajectories from scratch while continuously maintaining an external oracle state.

Experiments in the default WebArena Qwen and GPT configurations show that restoring oracle context recovers fewer than 9% of failed trajectories, while resetting the environment recovers 17–36%. The remaining 58–74% of failures persist even after both context and environment are restored; separately, human annotators label approximately 79% of failures as capability-caused. A continuously maintained oracle context improves results by 19 to 31 percentage points, and the gap between externally maintained and agent-generated state increases from 7% to 18% as tasks lengthen.

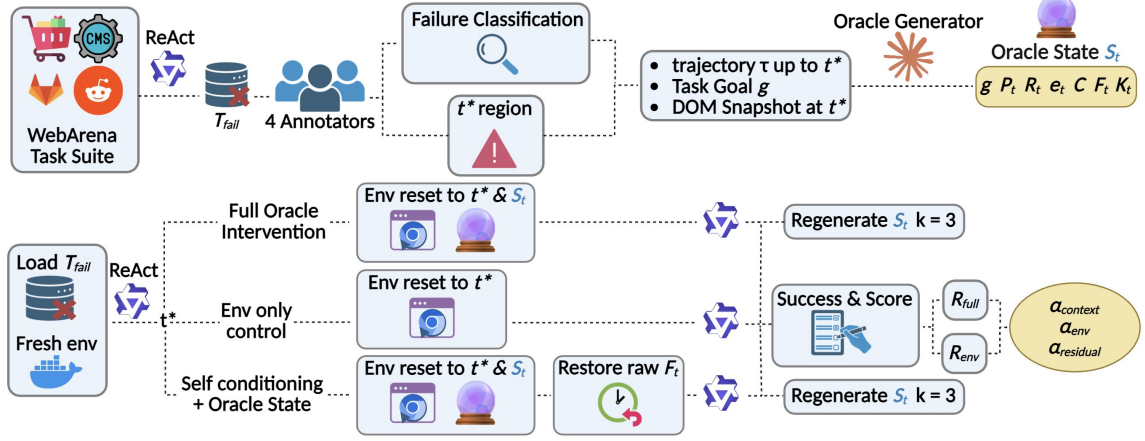


Figure 1: Oracle-state construction and intervention workflow. The top row shows how failed WebArena trajectories are annotated and converted into oracle states; the bottom row shows how the same task is replayed under full-oracle and environment-only interventions to estimate recovery and the alpha decomposition.

## 2 Related Works

Recent work shows that LLM agents degrade consistently on extended tasks as trajectory length increases, with models falling to below 50% accuracy after 15 steps due to self-conditioning on their own errors (Sinha et al., 2025), and highly capable agents reaching only 12% success on desktop tasks compared to 72% human performance (Xie et al., 2024). Web and browser-control benchmarks expose this problem directly (Yao et al., 2022; Deng et al., 2023; Zhou et al., 2024), while desktop and software-engineering benchmarks show similar pressure outside websites (Xie et al., 2024; Jimenez et al., 2024). Work on tool use, action-reasoning, memory, reflection, planning, and multi-agent interaction gives agents the machinery for multi-step execution (Yao et al., 2023; Qin et al., 2024; Packer et al., 2023; Shinn et al., 2023; Zhou et al., 2024; Wu et al., 2023; Wang et al., 2023; Park et al., 2023). Recent surveys and benchmark papers summarize these trends at the level of autonomous agents and memory systems (Wang et al., 2023; Liu et al., 2023; Zhang et al., 2024).

The above-mentioned findings have resulted in numerous methods for context management, which include learned compression (Kang et al., 2025), heuristic-based trajectory pruning (Xiao et al., 2025), observation masking (Lindenbauer et al., 2025), structured self-created state (Rozanov and Rei, 2024), and persistent external memory (Packer et al., 2023; Xu et al., 2025). Telling bad context apart from weak reasoning calls for intervention, not observation (Cemri et al., 2025; Zhu et al., 2025; Lu et al., 2025; Zhang et al., 2025). The closest works on this topic either use oracle ground-

ing to isolate planning from perception (Shlomov et al., 2025) or DoVer (Ma et al., 2025), which intervenes on agent instructions to debug multi-agent coordination issues. It remains that neither isolates context quality in single-agent systems.

Heuristic filtering shows that agents can tolerate context removal, and simple observation methods can match LLM-based summarization (Xiao et al., 2025; Lindenbauer et al., 2025). Our oracle interventions test the opposite direction: what happens when an agent is given strictly better context, rather than less context.

## 3 Methodology

### 3.1 Overview

We isolate context quality as a causal factor in agent failure in our methodology. Agents are evaluated under multiple contextual conditions centered on an oracle state  $S_t$ , which is a structured representation of the task state at step  $t$ . Because  $S_t$  is generated with privileged access to the recorded environment state, it serves as a ground-truth replacement for the agent’s noisy trajectory.

### 3.2 Oracle State Generation

An oracle state  $S_t$  at step  $t$  of an agent trajectory is defined as the septuple given below.

$$S_t = (g, P_t, R_t, e_t, C, F_t, K_t), \quad (1)$$

In  $S_t$ ,  $g$  denotes the task goal;  $P_t$  represents the set of tasks completed at or before step  $t$ ;  $R_t$  indicates the set of remaining tasks to achieve goal  $g$ ;  $e_t$  provides the model’s description of the environment at step  $t$ ;  $C$  specifies the set of constraints the agent must follow;  $F_t$  contains each failed attempt

up to step  $t$ ; and  $K_t$  includes key facts from the trajectory that are relevant to  $R_t$  overall.

If a subtask is completed before step  $t$ , it remains in  $P_t$  even if the agent has already failed.  $e_t$  and the cause field of each  $F_t$  entry must be supported by evidence from the true environment, rather than the agent providing its own belief.

Oracle states  $S_t$  are generated by prompting Claude Sonnet 4.6 with three inputs: the task goal  $g$ , the trajectory up to step  $t$  (which are represented as thought, action, observation triples), and the environment observation.

For Experiment 1 interventions, a single  $S_t$  is generated for each failed trajectory at the critical failure region  $t^*$ . In Experiments 2 and 3, five  $S_t$  are generated per trajectory at steps  $\{1, \lfloor 0.25L \rfloor, \lfloor 0.50L \rfloor, \lfloor 0.75L \rfloor, L\}$ , with  $t^*$  included if it does not coincide with any of these steps.

Human validation of the oracle samples was conducted on a random sample of 120 oracle state. Each  $S_t$  field and  $t^*$  region was manually evaluated across seven fields using a three-point scale (correct, partial, incorrect) against both the recorded trajectory and the WebArena task configuration, by two independent human annotators.

### 3.3 Experiment 1: Oracle State Intervention

Let  $T = \{\tau_1, \dots, \tau_N\}$  denote a set of agent trajectories on multi-step tasks, and let  $T_{\text{fail}} \subseteq T$  represent the subset of failed trajectories. For each failed trajectory  $\tau_i$ , the critical failure region  $t_i^*$  is defined as the region at which the agent takes an action from which it never recovers. This region must satisfy two criteria: (a) the action at step  $t$  is incorrect according to the ground-truth DOM, and (b) the agent does not return to a correct state in future steps.

Once the  $t^*$  classification and the oracle state at step  $t^*$  have been confirmed, the primary task agent (Qwen3.5-72B) is rerun on the failed trajectory  $\tau$ . When the agent arrives at step  $t_i^*$ , two conditions are applied to the same task through the WebArena evaluation harness’s intervention mode.

**Full oracle intervention.** The agent’s accumulated context is replaced with  $S_{t_i^*}$ , and execution resumes from the correct WebArena environment state at that step.

**Environment-only control.** The agent retains its original raw trajectory context  $\tau_{1:t_i^*}$ , but its environment is reset to its correct state at  $t_i^*$ ,

which is pulled from the WebArena Docker. The environment-only control isolates the effect of context by keeping the environment fixed.

Let  $Y_i^{\text{full}}$  and  $Y_i^{\text{env}}$  indicate whether trajectory  $\tau_i$  succeeds after the full-oracle and environment-only interventions. Recovery rates are

$$\begin{aligned} R_{\text{full}} &= \frac{1}{|T_{\text{fail}}|} \sum_{\tau_i \in T_{\text{fail}}} Y_i^{\text{full}}, \\ R_{\text{env}} &= \frac{1}{|T_{\text{fail}}|} \sum_{\tau_i \in T_{\text{fail}}} Y_i^{\text{env}}. \end{aligned} \quad (2)$$

$$\begin{aligned} \alpha_{\text{context}} &= R_{\text{full}} - R_{\text{env}}, \\ \alpha_{\text{env}} &= R_{\text{env}}, \\ \alpha_{\text{residual}} &= 1 - R_{\text{full}}. \end{aligned} \quad (3)$$

High  $\alpha_{\text{context}}$  supports the assumption underlying much of the context management research; high  $\alpha_{\text{env}}$  suggests that environment-recovery mechanisms are more important; high  $\alpha_{\text{residual}}$  means failures remain even after the context and environment interventions. This residual bucket is not itself a direct capability label: it can include capability limits, oracle-state mistakes, intervention-timing errors, environment irreversibility, replay artifacts, or failures to use the injected state format. We interpret it as consistent with capability limitations only alongside the human annotations, and we probe possible oracle bias by restricting decomposition to fully-correct oracle states.

Additionally,  $S_t$  intervention is tested at 25%, 50%, and 75% of the trajectory length, irrespective of the actual error location.

### 3.4 Critical-Failure-Region Annotation and Human Failure Attribution

For each failed trajectory we also collect human annotations of both  $t^*$  and failure classification, using four independent annotators with two assigned per task. When annotators disagree, we run the intervention at both labeled steps and report results separately, reporting later  $t^*$  as primary. For each step, annotators could navigate through the full failed trajectory and view the WebArena accessibility tree, action, thought, and raw output.

Annotators label a failure context-caused when the information given to the agent was wrong or incomplete and led it astray, and capability-caused when the information was correct but the agent acted on it incorrectly. For each trajectory, annotators record  $t^*$ , whether the failure was context-caused or capability-caused, and a brief justifica-

tion. We report inter-annotator agreement for failure classification through raw agreement, Cohen’s Kappa, and Gwet’s AC1. For  $t^*$ , we report exact matches between annotators as well as agreement between one and three steps.

### 3.5 Experiment 2: Performance-Length Scaling

Tasks are binned by trajectory length ( $\leq 10$ , 11-20, 21-40, 41-80 steps), and raw and  $S_t$  performance are compared within each bin.  $\Gamma(L)$ , the context gap, is defined by the following equation, where  $L$  denotes the length bin.

$$\Gamma(L) = \text{Acc}_L(\text{oracle}) - \text{Acc}_L(\text{raw}), \quad (4)$$

If context degradation matters,  $\Gamma(L)$  should sit near 0 for short trajectories and grow for long ones.

### 3.6 Experiment 3: Self-Generated vs External State

An agent’s own context maintenance may degrade significantly as trajectory length increases, since, as tasks get longer, agents become worse at summarizing their own context because it is already confused. We run three independent sets of full trajectories on the same tasks under a different method of context management.

**Raw trajectory.** The agent is provided its full unmodified trajectory at each step. No filtering, summarization, retrieval, or external state is applied in this condition.

**Self-generated state.** At each step, the agent generated its own summary of its state before action (Rozanov and Rei, 2024).

**Oracle external state.**  $S_t$  is regenerated every  $k$  steps under our previous definition of the state, with the LLM receiving its ground-truth environment state at each regeneration point. Between these points, we set  $k = 3$  by default and also test  $k = 1, 5, 10$  as sensitivity parameters.

Each of these interventions produces complete trajectories, and for each condition, we measure task success rate as a function of trajectory length.

We define the self-synthesis degradation gap as:

$$\Delta_{\text{synth}}(t) = \text{Acc}(S_t^{\text{oracle}}) - \text{Acc}(S_t^{\text{self}}), \quad (5)$$

where  $\text{Acc}(\cdot)$  denotes the success rate under the given condition, computed within trajectory bins. If  $\Delta_{\text{synth}}(t)$  increases with  $t$ , it confirms that the self-generated state degrades as the agent accumulates errors.

### 3.7 Evaluation Harness

An evaluation harness streamlines the WebArena environment, the task agent, and the context policy for each condition across all three experiments. A condition is defined by its agent variant and context policy. The agent variant helps determine the agent’s architecture, which is between ReAct (Yao et al., 2023) and StateAct (Rozanov and Rei, 2024). The context policy is a function that uses specific information the agent has accessed in prior steps. It does not alter the current observation or external data. In rollout mode, the agent starts at the WebArena start URL and continues until reaching a final state. In intervention mode, the harness opens a new environment configured for the task within trajectory  $t$  and replays the stored actions. If the loaded page URL ever diverges from the saved one, the run halts with the reason `replay_desync` and is excluded. Every failed trajectory that was not crashed or excluded due to `replay_desync` is finally evaluated by a success indicator and an overall score, along with  $R_{\text{full}}$ ,  $R_{\text{env}}$ , and the decomposition values.

The number of evaluated trajectories across conditions varies because analyses use different execution modes and exclusion criteria. As a result, we report  $N$  separately for each condition after excluding runs that crash, fail to replay, or are removed due to `replay_desync`.

### 3.8 Ablations

#### 3.8.1 Field Ablations

To determine the location of the recovery rate within  $S_t$ , the full intervention at  $t^*$  is repeated with each field removed from  $S_t$  that was provided to the agent. List fields ( $P_t$ ,  $R_t$ ,  $C$ ,  $F_t$ ,  $K_t$ ) are emptied, while string fields ( $g$ ,  $e_t$ ) are replaced with “[redacted]”. The decrease in  $R_{\text{full}}$  for each removed field is then computed using the same Experiment 1 pipeline.

#### 3.8.2 Regeneration Frequency

We test the sensitivity of oracle-external performance to the regeneration interval,  $k$ . In Experiment 3, the  $S_t$  is regenerated every  $k \in \{1, 3, 5, 10\}$  steps. This measures whether the benefits of the oracle-external state depend on the frequency of updates or remain stable when the state is not refreshed often.

Table 1: Experiment 1 Alpha Decomposition with 95% CI half-widths shown as  $\pm$  values (Wilson for  $\alpha_{\text{env}}$ ,  $\alpha_{\text{residual}}$ ; Wald for  $\alpha_{\text{context}}$ ).

| Setting                                 | N   | $R_{\text{full}}$ | $R_{\text{env}}$ | $\alpha_{\text{context}}$ | $\alpha_{\text{residual}}$ |
|-----------------------------------------|-----|-------------------|------------------|---------------------------|----------------------------|
| Qwen primary (later $t^*$ region)       | 120 | 28.1%             | 19.4%            | $8.7\% \pm 6.1$           | $71.9\% \pm 7.85$          |
| Qwen upper bound (earlier $t^*$ region) | 60  | 30.2%             | 17.6%            | $12.6\% \pm 9.4$          | $69.8\% \pm 11.5$          |
| Qwen pooled                             | 160 | 27.4%             | 21.3%            | $6.1\% \pm 4.55$          | $72.6\% \pm 6.75$          |
| GPT-5.2                                 | 100 | 41.8%             | 35.6%            | $6.2\% \pm 6.3$           | $58.2\% \pm 9.55$          |
| High-confidence $t^*$ region            | 80  | 25.8%             | 20.5%            | $5.3\% \pm 5.75$          | $74.2\% \pm 9.55$          |
| Verified-correct oracle subset          | 86  | 30.2%             | 21.7%            | $8.5\% \pm 7.0$           | $69.8\% \pm 9.7$           |

### 3.9 Model Generalizability

GPT-5.2 is used as an additional task agent in Experiment 1, which includes an oracle intervention in both the full and environment-only conditions on a subset of 100 tasks. The oracle generator comes from a different model family than the task agents, which helps prevent the oracle from simply predicting what another model in the same LLM family does, rather than what is actually correct. If Experiment 1 results are similar across both models, then the findings apply beyond Qwen3.5-27B. Experiments 2 and 3 don’t use GPT-5.2, since the main argument for the decomposition methodology relies on Experiment 1.

### 3.10 Baselines

We compare across five context-compression baselines: Sliding Window, ACON, AgentDiet, Observation masking, and Perfect Retrieval. Sliding window keeps only the most recent  $W$  tokens from the trajectory. ACON compresses the agent’s context, AgentDiet removes redundant trajectory parts, and observation masking keeps agent reasoning while replacing older tool outputs with placeholders. Perfect Retrieval embeds all trajectory chunks and retrieves the top- $k$  chunks using the ground-truth next action as the query, serving as an upper bound for context-retrieval methods.

## 4 Results

### 4.1 Experiment 1: Oracle State Intervention

Experiment 1 tests  $S_t$  intervention at the critical failure region  $t^*$  under both full oracle intervention and environment-only control. On 120 Qwen3.5-27B trajectories,  $\alpha_{\text{context}}$  was 8.7% [2.9, 15.1],  $\alpha_{\text{env}}$  19.4% [13.2, 27.4], and  $\alpha_{\text{residual}}$  71.9% [63.4, 79.1], demonstrating that the task agent still failed to recover even with the injection of full oracle context (Table 1). Four robustness checks test whether this

result depends on  $t^*$  selection: earlier  $t^*$ , pooled, fixed-fraction intervention, and high-confidence  $t^*$  region (with within-three-step agreement of 0.93). Restricting to 86 verified states yields  $\alpha_{\text{context}} = 8.5\%$ , consistent with the primary 8.7%, supporting context replacement at the annotated region rarely rescues already-failed agents.

### 4.2 Convergent Validation: Human Failure Attribution

We use human failure attribution as an independent check to validate the decomposition results. Final human attribution assigns only 11.7% of failed trajectories to context, compared to 79.1% capability-caused labels, with 9.2% disagreement. Annotators generally agreed on broad classification, with raw agreement of 88.1% across 240 labels over 120 tasks, Gwet’s AC1 between 0.75 and 0.8, and Cohen’s  $\kappa$  between 0.50 and 0.58. However, exact agreement on the critical failure region  $t^*$  (0.44 to 0.49) is lower than agreement within three steps (0.86 to 0.90).

### 4.3 Baselines

All of the context-management baselines outperform raw trajectory prompting, which achieves only 23.4% success. Sliding Window and observation masking improved to 39.6% and 38.9%, while ACON and AgentDiet reached 43.7% and 42.8%, respectively. Perfect Retrieval performs best among the baselines at 48.6%, but this condition uses the ground-truth next action and represents an ideal upper bound. Still, oracle-external state reaches 51.2%, suggesting that  $S_t$  adds a structured representation of progress, remaining sub-tasks, and the environment state beyond retrieval over past tokens.

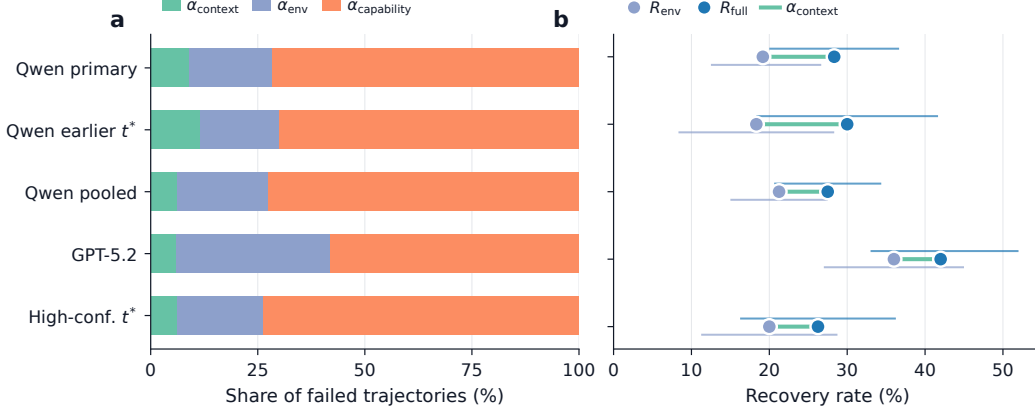


Figure 2: Three-way decomposition of multi-step agent failures across intervention settings. Residual unrecovered failures dominate in every setting, while the context-attributable component remains small relative to environment recovery.

Table 2: Overall Success Rate of all conditions.

| Method              | N   | Success Rate |
|---------------------|-----|--------------|
| Raw                 | 310 | 23.4%        |
| Self-generated      | 260 | 41.8%        |
| Oracle-external     | 255 | 51.2%        |
| Sliding Window      | 275 | 39.6%        |
| Observation Masking | 270 | 38.9%        |
| ACON                | 255 | 43.7%        |
| AgentDiet           | 250 | 42.8%        |
| Perfect Retrieval   | 245 | 48.6%        |

#### 4.4 Experiment 2: Context Gap $\Gamma(L)$

Experiment 2 uses  $\Gamma(L)$ , the difference between oracle-external success and raw success per length bin, to indicate how  $S_t$  changes success across trajectory lengths. Table 3 shows that the gap is positive in every bin: +18.9 points for tasks of at most 10 steps and +28.6 points in the longest 41–80-step bin. Raw success falls from 37.5% to 2.9% across these bins, while oracle-external success remains 31.5% in the longest bin, supporting the interpretation that  $S_t$  is most valuable as a preventive intervention as rollouts lengthen.

#### 4.5 Experiment 3: Self-Generated vs Oracle-External State ( $\Delta_{\text{synth}}(t)$ )

Experiment 3 evaluates whether agents can reliably maintain their own states or whether the state needs to be maintained externally. Table 2 shows that raw trajectory prompting had a 23.4% success rate, self-generated state increased to 41.8%, and oracle-external state achieved 51.2%. On the matched 210-task subset in Table A14, self-generated state achieved 43.6% success while oracle-external state achieved 54.9%, an 11.3-point gain for externally managed state. The self-synthesis degradation gap rises from 7.4% for trajectories of 10 steps or fewer

Table 3: Context gap  $\Gamma(L)$  and synthesis gap  $\Delta_{\text{synth}}(t)$  by trajectory length.

| Length Bin | N   | Raw   | Oracle-ext. | $\Gamma(L)$ | $\Delta_{\text{synth}}(t)$ |
|------------|-----|-------|-------------|-------------|----------------------------|
| $\leq 10$  | 120 | 37.5% | 56.4%       | +18.9%      | +7.4%                      |
| 11–20      | 60  | 9.8%  | 36.5%       | +26.7%      | +12.1%                     |
| 21–40      | 50  | 5.6%  | 36.4%       | +30.8%      | +10.6%                     |
| 41–80      | 60  | 2.9%  | 31.5%       | +28.6%      | +17.8%                     |

to 17.8% in the 41–80-step bin. Thus self-generated state is useful, improving success by 18.4 points over raw prompting, but still trails state maintained outside the agent’s own error-prone reasoning process.

#### 4.6 Model Generalizability

To test whether decomposition is specific to Qwen3.5-27B, we rerun Experiment 1 with GPT-5.2 on a matched subset of 100 tasks. GPT-5.2 achieves higher recovery rates with a  $R_{\text{full}} = 41.8\%$  and  $R_{\text{env}} = 35.6\%$ . However, its fraction of failures due to context remains small with  $\alpha_{\text{context}} = 6.2\%$  while  $\alpha_{\text{residual}} = 58.2\%$ . In a matched model comparison, GPT-5.2 improves oracle recovery by 24.1 points and environment-only recovery by 19 points, but  $\alpha_{\text{context}}$  stays low for both models.

#### 4.7 Preliminary Cross-Benchmark Generalization

To test whether decomposition is specific to WebArena, we evaluate the oracle-state intervention on a 50-task subset on Terminal-Bench (Merrill et al., 2026). Two annotators manually looked over 38 Terminal-Bench  $S_t$  and validated field and  $t^*$  correctness. Of the 50 tasks, 38 were selected for paired intervention. Full oracle recovery reached 42.1%, while environment-only recovery reached

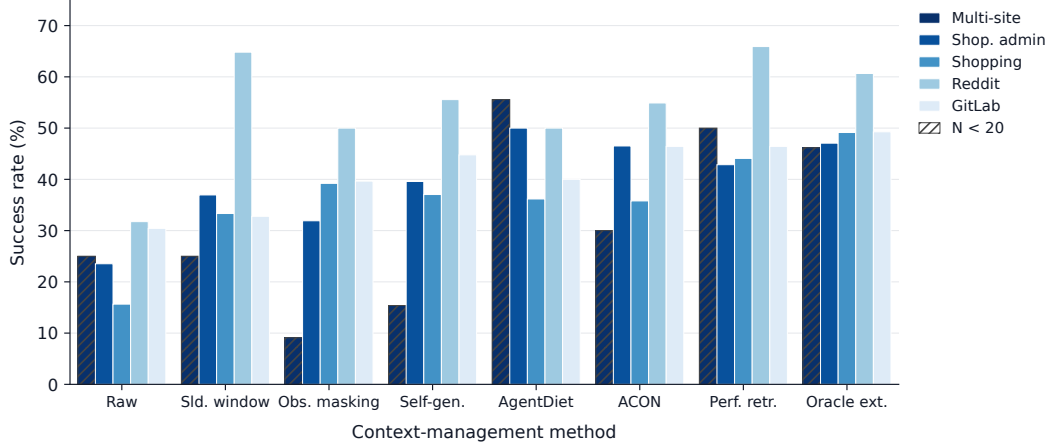


Figure 3: Success rate across WebArena sites and context-management methods. Hatched bars mark site-method cells with  $N < 20$  and should be interpreted as exploratory. Oracle-external is strongest overall, while perfect retrieval remains competitive, especially on some site subsets.

31.6%, ending with  $\alpha_{\text{context}} = 10.5\%$ ,  $\alpha_{\text{env}} = 31.6\%$ , and  $\alpha_{\text{residual}} = 57.9\%$ . These observations mirror the pattern observed on WebArena where context replacement explains only a small number of failures while residual unrecovered failures remain the largest.

## 4.8 Ablations

### 4.8.1 Field Ablations

In Field Ablations, the full  $t^*$  intervention was repeated with each of the seven oracle fields in the oracle state removed. List fields ( $P_t, R_t, C, F_t, K_t$ ) were emptied and string fields ( $g, e_t$ ) were replaced with [redacted]. In Table A5, removing any field decreases  $R_{\text{full}}$ . Both the environment state (-7.6%) and the remaining subtasks (-6.9%) had the highest drop in full oracle recovery rate, whereas failed attempts (-1.8%), constraints (-1.6%), and goal (-0.4%) had the lowest drop in full oracle recovery rate, revealing that the recovery signal concentrates in fields that describe the agent’s current situation and not fields that describe the past. The -0.4% for the goal is expected, since the agent already has it in its base prompt. These results indicate that when the oracle context helps recover agents from multi-step failures, it is mostly due to the environment state and remaining subtasks.

### 4.8.2 Regeneration Frequency Ablations

In Regeneration Frequency ablations, the oracle state is regenerated every  $k$  steps, with  $k$  set to 1, 3, 5, or 10. In Table A6, success rises from 45.7% at  $k = 10$  to 51.6% at  $k = 1$ , while  $k = 3$  reaches 51.2%. This supports  $k = 3$  as the default and suggests that  $k = 10$  leaves the agent with an

increasingly outdated state.

## 4.9 Perfect Retrieval vs. Structured Oracle State

Perfect Retrieval provides an oracle upper bound for retrieval over trajectory history by embedding trajectory chunks and retrieving top- $k$  chunks using the ground-truth next action as the query. Even with this privileged access, Perfect Retrieval reaches 48.6%, while oracle-external state reaches 51.2%, showing that structured state helps beyond access to relevant past tokens by organizing history into variables the agent can act on.

## 4.10 Discussion

Our results show that context quality is a key component of success for multi-step agents, but not in the way assumed by previous literature. Results of the oracle intervention experiment show that  $\alpha_{\text{context}}$  stays low across settings, indicating that replacing an agent’s context at the critical failure region recovers only a small fraction of failed trajectories. However, the oracle-external state significantly improves upon raw trajectory prompting, with the oracle-external state reaching 51.2% success compared to the 23.4% of raw trajectory prompting. This contrast indicates that context management is primarily preventive, with state interventions reducing the likelihood that the agent accumulates errors over the course of a trajectory, rather than fixing an already error-prone agent. The self-generated-state result sharpens this point: self-generated state improves over raw prompting, but still trails oracle-external state by 11.3 points on the matched subset, and the gap grows on longer trajec-



Figure 4: Raw and oracle-external success as a function of trajectory length. Raw performance collapses with length, while oracle-external state preserves substantially more performance on longer trajectories.

tories. The decomposition also suggests that engineering the context alone will not solve multi-step agent failure. Residual unrecovered failures dominate across settings, reaching 71.9% for the primary Qwen3.5-27B setting and 58.2% for GPT-5.2. Together with the human annotations, these results are consistent with capability-related limits, but  $\alpha_{\text{residual}}$  should not be read as a pure capability measure. Even strong information about the state does not guarantee correct execution, since agents can still improve in key traits such as stronger planning, action selection, and recovery. This also highlights an often-overlooked source of failure: agents can damage the environment as well as their own memory. Environment-only recovery accounts for a non-trivial portion of recovered failures; agent systems may need explicit mechanisms for environment recovery before the agent does irreversible operations. The field ablation gives a concrete memory-design lesson: the largest recovery drops come from removing environment state and remaining subtasks, while removing the original goal barely matters. Useful agent memory should therefore track current state and remaining work, not only compress the full history or restate the task.

## 5 Conclusion

We presented an oracle-state intervention framework for testing whether multi-step agent failures are caused by context or other residual factors after context and environment correction. Across 420 WebArena trajectories,  $\alpha_{\text{context}}$  stays low across

Qwen3.5-27B and GPT-5.2, while residual unrecovered failures account for 71.9% and 58.2% in the respective settings. Human annotations support a capability-related interpretation of this residual, with 79.1% of final failures labeled capability-caused compared to 11.7% context-caused. However, continuous oracle-external state improves success throughout the trajectory and outperforms self-generated state as trajectory length increases. Context management is therefore not irrelevant; its main value lies in preventing error accumulation rather than saving an agent after it reaches a failure point.

## Limitations

Experiment 1 is not powered to resolve single-digit  $\alpha_{\text{context}}$  differences between settings.  $S_t$  is not perfect since it is still generated by an LLM, which can omit or misrepresent parts of the environment, despite strong validation. Our experiments are primarily conducted in WebArena navigation tasks, with a smaller generalization check through Terminal-Bench. Since the Terminal-Bench subset only includes 38 paired interventions, validation across broader environments remains necessary. Identifying the exact critical failure region  $t^*$  is difficult, as annotators often agree on a window in which  $t^*$  falls but not on its exact location. The intervention is applied at an annotated point in the critical failure region, so Experiment 1 should be interpreted as a post-failure rescue test rather than a complete estimate of context’s preventive value. The resid-

ual bucket can also include oracle mistakes, timing errors, replay artifacts, environment irreversibility, or failures to use the injected state. Our oracle-external condition should be viewed as an upper bound, rather than a deployable method, since it requires privileged access to the environment state.

## **Ethical Considerations**

Our experiments are conducted in controlled benchmark environments, and we do not use private user data or real user accounts. Our oracle-state intervention should be used as a diagnostic tool for understanding agent failures, not as a monitoring system. More reliable agents can improve automation but can also enable harmful online actions if proper constraints are not implemented. We support safeguards such as constrained action spaces, human oversight, reversible operations, and environmental recovery mechanisms.

## References

- M. A. Merrill, A. G. Shaw, and others. 2026. Terminal-Bench: Benchmarking Agents on Hard, Realistic Tasks in Command Line Interfaces. *arXiv preprint arXiv:2601.11868*.
- M. Cemri, M. Z. Pan, S. Yang, and others. 2025. Why do multi-agent LLM systems fail? *arXiv preprint arXiv:2503.13657*.
- M. Ma and others. 2025. DoVer: Intervention-driven auto debugging for LLM multi-agent systems. *arXiv preprint arXiv:2512.06749*.
- X. Deng, Y. Gu, B. Zheng, S. Chen, S. Stevens, B. Wang, H. Sun, and Y. Su. 2023. Mind2Web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- M. Kang, W.-N. Chen, D. Han, and others. 2025. ACON: Optimizing context compression for long-horizon LLM agents. *arXiv preprint arXiv:2510.00615*.
- M. Lindenbauer and others. 2025. The complexity trap: Simple observation masking is as efficient as LLM summarization for agent context management. In *NeurIPS DL4Code Workshop*.
- R. Lu, Y. Li, and Y. Huo. 2025. Exploring autonomous agents: A closer look at why they fail when completing tasks. *arXiv preprint arXiv:2508.13143*.
- X. Liu, H. Yu, H. Zhang, and others. 2023. Agent-Bench: Evaluating LLMs as agents. *arXiv preprint arXiv:2308.03688*.
- N. Rozanov and M. Rei. 2024. StateAct: State tracking and reasoning for acting and planning with large language models. *arXiv preprint arXiv:2410.02810*.
- C. Packer, S. Wooders, K. Lin, V. Fang, S. G. Patil, I. Stoica, and J. E. Gonzalez. 2023. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*.
- Y. Qin, S. Liang, Y. Ye, and others. 2024. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations (ICLR)*.
- J. S. Park, J. O'Brien, C. J. Cai, and others. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*.
- S. Shlomov, N. Wiesel, E. Sela, O. Levy, R. Galanti, and R. Abitbol. 2025. From grounding to planning: Benchmarking bottlenecks in web agents. In *European Conference on Artificial Intelligence (ECAI)*.
- N. Shinn, F. Cassano, E. Berman, A. Gopinath, K. Narasimhan, and S. Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- M. Sinha and others. 2025. The illusion of diminishing returns: Measuring long-horizon execution in LLMs. *arXiv preprint arXiv:2509.09677*.
- T. Xie and others. 2024. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- C. E. Jimenez, J. Yang, A. Wettig, and others. 2024. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations (ICLR)*.
- W. Xu, Z. Liang, K. Mei, and others. 2025. A-MEM: Agentic memory for LLM agents. *arXiv preprint arXiv:2502.12110*.
- S. Yao, J. Zhao, D. Yu, and others. 2023. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.
- S. Yao, H. Chen, J. Yang, and K. Narasimhan. 2022. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- L. Wang, C. Ma, X. Feng, and others. 2023. A survey on large language model based autonomous agents. *arXiv preprint arXiv:2308.11432*.
- G. Wang, Y. Xie, Y. Jiang, and others. 2023. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.
- Q. Wu, G. Bansal, J. Zhang, and others. 2023. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*.
- Z. Zhang, X. Bo, C. Ma, R. Li, X. Chen, Q. Dai, and others. 2024. A survey on the memory mechanism of large language model based agents. *arXiv preprint arXiv:2404.13501*.
- Y.-A. Xiao and others. 2025. Reducing cost of LLM agents with trajectory reduction. *arXiv preprint arXiv:2509.23586*.
- A. Zhou, K. Yan, M. Shlapentokh-Rothman, H. Wang, and Y.-X. Wang. 2024. Language agent tree search unifies reasoning acting and planning in language models. In *International Conference on Machine Learning (ICML)*.
- S. Zhang, M. Yin, J. Zhang, and others. 2025. Which agent causes task failures and when? On automated failure attribution of LLM multi-agent systems. *arXiv preprint arXiv:2505.00212*.
- S. Zhou, F. F. Xu, H. Zhu, and others. 2024. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations (ICLR)*.
- K. Zhu, Y. Liu, H. Li, and others. 2025. Where LLM agents fail and how they can learn from failures. *arXiv preprint arXiv:2509.25370*.

## A Experimental Setup

Table A1: Experimental setup.

| Component                                   | Details                                                                                                                         |
|---------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| Primary task agent                          | Qwen3.5-27B                                                                                                                     |
| Generalizability agent                      | GPT-5.2 (100-task subset, Experiment 1 only)                                                                                    |
| Oracle generator                            | Claude 4.6 Sonnet with ground-truth environment access                                                                          |
| Benchmarks                                  | WebArena; Terminal-Bench generalization subset                                                                                  |
| Task categories                             | Shopping, Shopping Admin, GitLab, Reddit, Multi-site                                                                            |
| Trajectory collection                       | 420 failed trajectories (ReAct agent), primary agent                                                                            |
| Oracle validation                           | 120 human-verified oracle states, field-level agreement; 38 Terminal-Bench oracle states                                        |
| Failure annotation                          | 120 failed trajectories, 240 labels, two annotators per task                                                                    |
| <i>Context conditions (Experiments 1–3)</i> |                                                                                                                                 |
| Raw trajectory                              | Full unmodified agent history                                                                                                   |
| Self-generated state                        | Agent produces chain-of-state summary                                                                                           |
| Oracle external state                       | Frontier-LLM-generated $S_t$ with environment access                                                                            |
| Environment-only control                    | Correct environment state + raw trajectory context                                                                              |
| <i>Baselines</i>                            |                                                                                                                                 |
| Sliding window                              | Last $W$ tokens of trajectory                                                                                                   |
| ACON                                        | Learned context compression                                                                                                     |
| AgentDiet                                   | Heuristic expired/redundant removal                                                                                             |
| Observation masking                         | Replace tool outputs with placeholder                                                                                           |
| Perfect retrieval                           | RAG over full trajectory with oracle-relevant query                                                                             |
| <i>Metrics</i>                              |                                                                                                                                 |
| Primary                                     | Task success rate, $R_{\text{full}}$ and $R_{\text{env}}$ (Eqs. 2)                                                              |
| Diagnostic                                  | $\alpha_{\text{context}}, \alpha_{\text{env}}, \alpha_{\text{residual}}, \Delta_{\text{synth}}(t)$ (Eq. 5), $\Gamma(L)$ (Eq. 4) |
| Efficiency                                  | Tokens consumed per condition                                                                                                   |

## B Tasks Source and Selection for WebArena and Terminal-Bench

The WebArena subset used here includes Shopping, Shopping Admin, GitLab, Reddit, and multi-site tasks. We randomly selected tasks (seed 42) from WebArena’s test split. Tasks requiring Wikipedia or Map were dropped.

For WebArena, trajectories were collected by running the task agent against the four live Docker environments, and session cookies were refreshed before the next trajectory to ensure the agent remained authenticated and prevent task failure due to expired credentials. For Terminal-Bench, we randomly sampled a 50-task subset (seed 42) from the evaluation split. Failed trajectories were collected using the same pipeline as that used for WebArena. The environment was fully reset between trajectories to prevent the agent from potentially using past compiled dependencies to cheat. A trajectory qualifies as a failed trajectory if its evaluator score is `success=False` and the stop reason is not `crash` or `parse_failures`.

## C Model Configurations and Inference Settings

Table A2: Model configurations and inference settings used across experiments.

| Component              | Setting                        | Value                                                          |
|------------------------|--------------------------------|----------------------------------------------------------------|
| Primary Task Agent     | Model                          | Qwen3.5-27B (qwen/qwen3.5-27b via OpenRouter)                  |
|                        | Temperature                    | 0.0                                                            |
|                        | Output max tokens              | 2048                                                           |
|                        | Top-p                          | 1.0                                                            |
|                        | Max steps per task attempt     | 50                                                             |
|                        | API timeout                    | 60 seconds per call                                            |
|                        | Max observation length         | 4096 tokens                                                    |
|                        | Thinking mode                  | Enabled                                                        |
| Oracle State Generator | Model                          | Claude Sonnet 4.6 (anthropic/claude-sonnet-4-6 via OpenRouter) |
|                        | Temperature                    | 0.0                                                            |
|                        | Output max tokens              | 4096                                                           |
|                        | Prompt caching                 | Enabled                                                        |
|                        | Default regeneration frequency | Every $k = 3$ steps                                            |
| Generalizability Agent | Model                          | GPT-5.2 (openai/gpt-5.2 via OpenRouter)                        |
|                        | Temperature                    | 0.0                                                            |
|                        | Max tokens                     | 2048                                                           |
|                        | Other settings                 | Same as the primary task agent                                 |

## D Critical-Failure-Region Annotation and Agreement Definition

Table A3: Inter-annotator agreement: failure classification.

| Annotators | Raw Agreement | Cohen’s $\kappa$ | Gwet’s AC1 |
|------------|---------------|------------------|------------|
| 1 vs 2     | 84.2%         | 0.58             | 0.80       |
| 3 vs 4     | 76.8%         | 0.51             | 0.75       |
| 1 vs 3     | 79.5%         | 0.49             | 0.77       |
| 2 vs 4     | 81.1%         | 0.54             | 0.78       |
| 1 vs 4     | 80.4%         | 0.53             | 0.77       |
| 2 vs 3     | 78.6%         | 0.50             | 0.74       |

Table A4: Inter-annotator agreement: critical failure region  $t^*$ .

| Annotators | Exact Match | Within 1 step | Within 3 steps |
|------------|-------------|---------------|----------------|
| 1 vs 2     | 0.49        | 0.76          | 0.90           |
| 3 vs 4     | 0.47        | 0.78          | 0.89           |
| 1 vs 3     | 0.44        | 0.73          | 0.86           |
| 2 vs 4     | 0.46        | 0.75          | 0.88           |
| 1 vs 4     | 0.45        | 0.74          | 0.87           |
| 2 vs 3     | 0.43        | 0.72          | 0.85           |

### D.1 $t^*$ Annotations

We define  $t^*$  as the earliest incorrect action from which the agent does not recover and never returns to a correct task state. Annotators identified this critical failure region for each failed trajectory. Steps in which the agent makes a mistake but later recovers and finishes the task subgoal are not marked as  $t^*$ . However, if a step causes the agent to enter a persistent failure mode, such as looping, and the agent never recovers, the step is marked as  $t^*$ .

Since trajectory failures often emerge gradually, with the critical action often not obvious, we treat  $t^*$  as a step-level annotation, but we also report agreement within local windows. Specifically, we report exact  $t^*$  agreement within  $\pm 1$  step and  $\pm 3$  steps. Annotators often agreed on the region for  $t^*$ , but not the exact step. This is shown by a lower exact-match agreement of 43%-49%, but a higher three-step agreement of 85%-90%.

### D.2 Failure-cause Labels

Annotators additionally assigned a cause label to each failed trajectory. We used the following definitions:

**Context-caused:** A failure was deemed context-caused when an agent’s incorrect action was attributable primarily to missing or incorrect information in its context.

**Capability-caused:** A failure was deemed capability-caused when all relevant information was available to the agent, but the agent still reasoned incorrectly, selected an invalid or unhelpful action, pursued the wrong subgoal, failed to use available evidence, or failed to recover from an earlier mistake.

## E Ablation Results

Table A5: Oracle Field Ablation.

| Removed Field                | Drop in $R_{\text{full}}$ |
|------------------------------|---------------------------|
| Environment state ( $e_t$ )  | -7.6%                     |
| Remaining subtasks ( $R_t$ ) | -6.9%                     |
| Key facts ( $K_t$ )          | -4.1%                     |
| Completed subtasks ( $P_t$ ) | -3.7%                     |
| Failed attempts ( $F_t$ )    | -1.8%                     |
| Constraints ( $C$ )          | -1.6%                     |
| Goal ( $g$ )                 | -0.4%                     |

Table A6: Regeneration Frequency Sensitivity.

| Condition                    | N   | Success |
|------------------------------|-----|---------|
| Oracle state, every 10 steps | 100 | 45.7%   |
| Oracle state, every 5 steps  | 100 | 49.4%   |
| Oracle state, every 3 steps  | 100 | 51.2%   |
| Oracle state, every 1 step   | 100 | 51.6%   |

## F Supplementary Tables

Table A7: Fixed-fraction intervention sensitivity.

| Intervention Point              | N   | $\alpha_{\text{context}}$ |
|---------------------------------|-----|---------------------------|
| 25% $L$                         | 90  | 9.6%                      |
| 50% $L$                         | 90  | 7.1%                      |
| 75% $L$                         | 90  | 4.8%                      |
| Human-annotated $t^*$ (primary) | 120 | 8.7%                      |

Table A8: Cross-Model Matched Comparison ( $N = 100$ ).

| Metric                    | GPT-5.2 – Qwen3.5-27B |
|---------------------------|-----------------------|
| Full-oracle recovery      | +24.1%                |
| Env-only recovery         | +19.0%                |
| $\alpha_{\text{context}}$ | +5.1%                 |

Table A9: Cross-Model Matched on GPT subset ( $N = 100$ ).

| Model       | $\alpha_{\text{context}}$ |
|-------------|---------------------------|
| Qwen3.5-27B | 1.1%                      |
| GPT-5.2     | 6.2%                      |

Table A10: GPT-5.2 recovery by trajectory length.

| Length Bin | N  | Full-oracle recovery | Env-only recovery | $\alpha_{\text{context}}$ |
|------------|----|----------------------|-------------------|---------------------------|
| $\leq 10$  | 35 | 48.2%                | 40.4%             | +7.8%                     |
| 11–20      | 25 | 41.1%                | 35.0%             | +6.1%                     |
| 21–40      | 20 | 36.4%                | 27.9%             | +8.5%                     |
| 41–80      | 20 | 29.7%                | 24.8%             | +4.9%                     |

Table A11: Context gap  $\Gamma(L)$ : oracle-external vs raw by length.

| Length Bin | N   | Raw success | Oracle-external success | $\Gamma(L)$ |
|------------|-----|-------------|-------------------------|-------------|
| $\leq 10$  | 120 | 37.5%       | 56.4%                   | +18.9%      |
| 11–20      | 60  | 9.8%        | 36.5%                   | +26.7%      |
| 21–40      | 50  | 5.6%        | 36.4%                   | +30.8%      |
| 41–80      | 60  | 2.9%        | 31.5%                   | +28.6%      |

Table A12: Synthesis gap  $\Delta_{\text{synth}}(t)$ : oracle-external vs self-generated by length.

| Length Bin | N   | $\Delta_{\text{synth}}(t)$ |
|------------|-----|----------------------------|
| $\leq 10$  | 120 | +7.4%                      |
| 11–20      | 60  | +12.1%                     |
| 21–40      | 50  | +10.6%                     |
| 41–80      | 60  | +17.8%                     |

Table A13: Overall Success Rate of all conditions.

| Method              | N   | Success Rate |
|---------------------|-----|--------------|
| Raw                 | 310 | 23.4%        |
| Self-generated      | 260 | 41.8%        |
| Oracle-external     | 255 | 51.2%        |
| Sliding Window      | 275 | 39.6%        |
| Observation Masking | 270 | 38.9%        |
| ACON                | 255 | 43.7%        |
| AgentDiet           | 250 | 42.8%        |
| Perfect Retrieval   | 245 | 48.6%        |

Table A14: Experiment 3 Matched Comparisons.

| Comparison                          | N   | Result |
|-------------------------------------|-----|--------|
| Self-generated matched              | 210 | 43.6%  |
| Oracle external matched             | 210 | 54.9%  |
| Oracle external over self-generated | 210 | +11.3% |

Table A15: Oracle State Validation Field Correctness.

| Benchmark      | N   | Adjudicated Correctness | Raw Inter-Validator Agreement | Cohen’s $\kappa$ | Gwet’s AC1 |
|----------------|-----|-------------------------|-------------------------------|------------------|------------|
| WebArena       | 120 | 87.90%                  | 88.70%                        | 0.56             | 0.79       |
| Terminal-Bench | 38  | 87.50%                  | 85.00%                        | 0.50             | 0.74       |

Table A16: Oracle State Validation  $t^*$  Region Correctness.

| Benchmark      | N   | Adjudicated Correctness | Raw Inter-Validator Agreement | Cohen’s $\kappa$ | Gwet’s AC1 |
|----------------|-----|-------------------------|-------------------------------|------------------|------------|
| WebArena       | 120 | 84.20%                  | 85.00%                        | 0.48             | 0.72       |
| Terminal-Bench | 38  | 82.50%                  | 82.50%                        | 0.43             | 0.68       |

Table A17: Terminal-Bench Oracle Intervention Alpha Decomposition.

| Setting                                 | N  | Full-Oracle Recovery | Env-Only Recovery | $\alpha_{\text{context}}$ | $\alpha_{\text{env}}$ | $\alpha_{\text{residual}}$ |
|-----------------------------------------|----|----------------------|-------------------|---------------------------|-----------------------|----------------------------|
| Terminal-Bench (primary)                | 38 | 42.10%               | 31.60%            | 10.50%                    | 31.60%                | 57.90%                     |
| Terminal-Bench (high-confidence $t^*$ ) | 24 | 50.00%               | 37.50%            | 12.50%                    | 37.50%                | 50.00%                     |

Table A18: Terminal-Bench Task Corpus.

| Quantity                                      | Value |
|-----------------------------------------------|-------|
| Task subset                                   | 50    |
| Failed trajectories selected for intervention | 38    |
| Paired intervention N                         | 38    |
| High-confidence $t^*$ subset N                | 24    |

## G Supplementary Figures

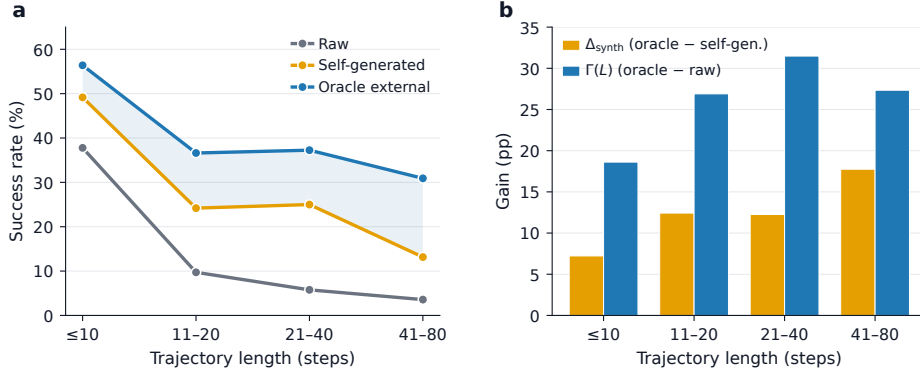


Figure A1: Prevention and synthesis gaps by trajectory length. The oracle-external advantage over self-generated state widens on longer trajectories.

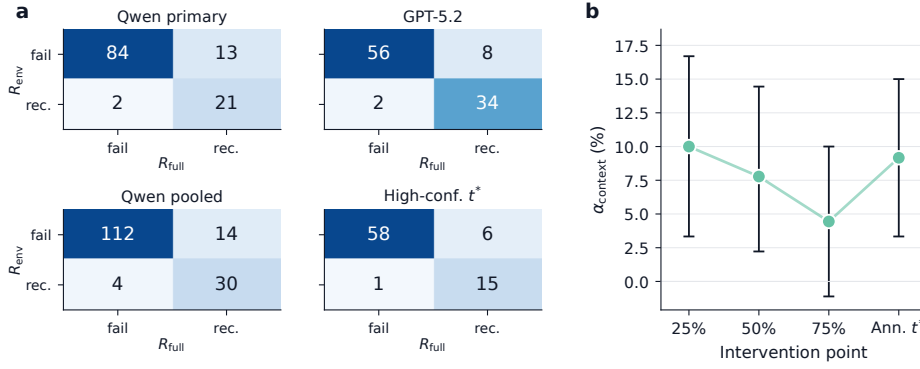


Figure A2: Recovery and timing sensitivity. Stronger models improve absolute recovery, but the full-oracle over environment-only gap remains limited.

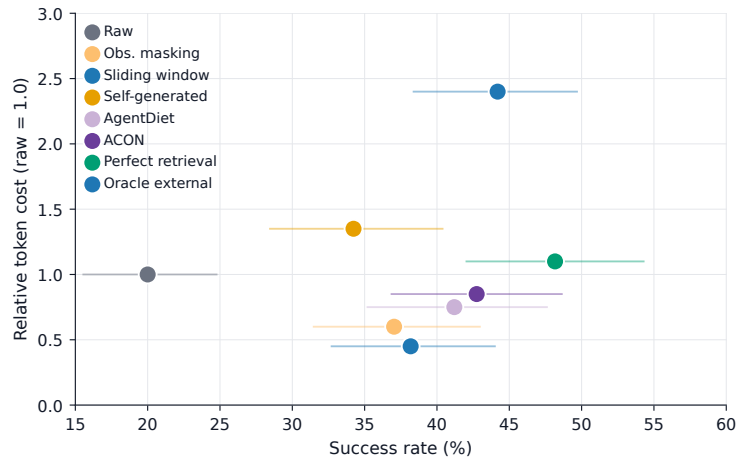


Figure A3: Regeneration-frequency sensitivity. Oracle-state success increases modestly as the state is refreshed more often, with  $k = 3$  and  $k = 1$  nearly tied.

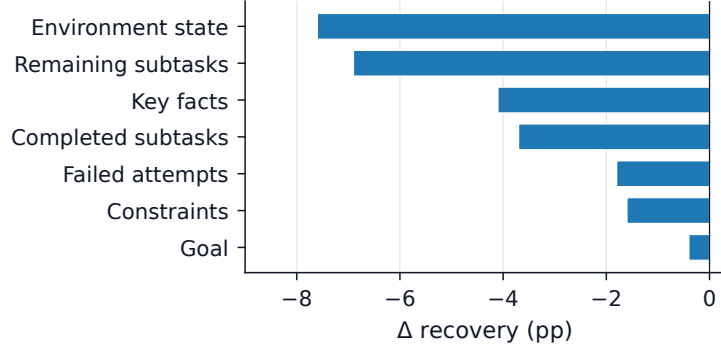


Figure A4: Per-field ablation of oracle state. Environment state and remaining-subtasks fields account for the largest drops in recovery when removed.

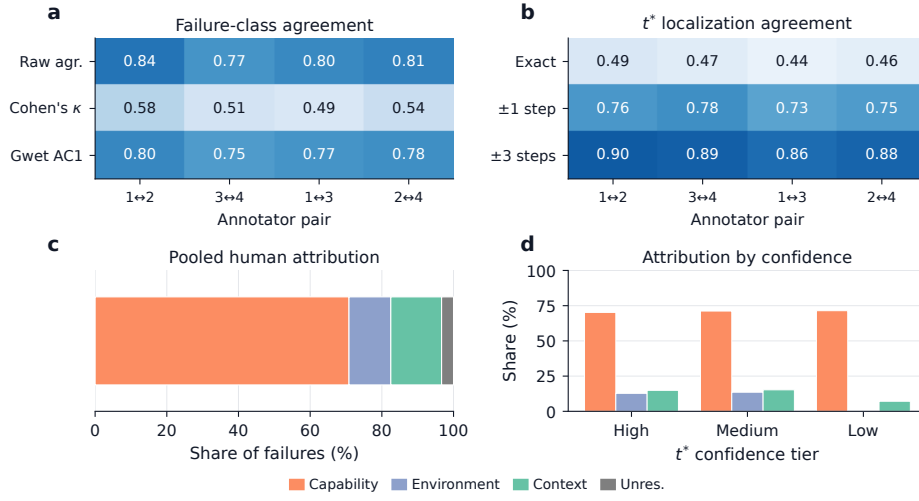


Figure A5: Annotation reliability and human attribution. Pairwise failure-class agreement and  $t^*$  agreement show that broad failure attribution is more stable than exact-step localization.

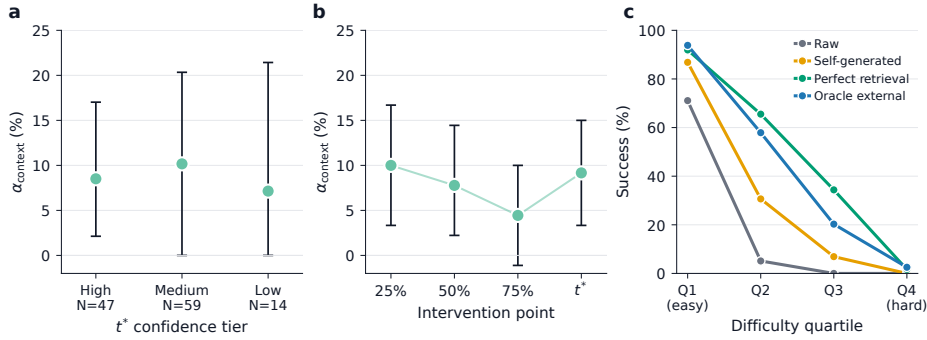


Figure A6: Robustness of  $\alpha_{\text{context}}$ . Context attribution remains in the same low range across primary, earlier, pooled, GPT-5.2, high-confidence, and fixed-fraction intervention settings.

## H Oracle State Generator Prompt

You are an oracle state generator for a web agent. Your job is to produce a structured, factually grounded summary of the agent's task state at a specific step in its trajectory.

The CURRENT step is {t}. Only reason about what has happened UP TO AND INCLUDING step {t}. Do not speculate about future steps.

TASK GOAL:  
{task\_goal}

TRAJECTORY UP TO STEP {t}:  
{trajectory\_text}

GROUND-TRUTH DOM AT STEP {t}:  
{dom\_snapshot}

Output a structured oracle state with exactly these fields:

- g (goal): restate the task goal clearly and completely
- P\_t (completed subtasks): list every subtask fully accomplished before or at step {t}
- R\_t (remaining subtasks): list what still needs to be done to complete the goal
- e\_t (environment state): describe the current page and state based ONLY on the ground-truth DOM, not the agent's observations (which may be stale or wrong)
- C (constraints): any active constraints the agent must respect going forward
- F\_t (failed attempts): list each failed action with its correctly attributed cause, grounded in DOM evidence not agent belief
- K\_t (key facts): important facts extracted from the trajectory useful for completing remaining subtasks

Critical rules:

- For e\_t: derive ONLY from the ground-truth DOM, not from the agent's observations.
- For F\_t: state the action, the step it occurred, and the cause per DOM evidence. Wrong: "button was unresponsive". Right: "Clicked Submit at step 8 – email field still empty per DOM".
- If a field has no content, return an empty list []. Do not invent values.
- Do not hallucinate. Use only what is in the trajectory and DOM.

Output valid JSON only. No preamble, no explanation, no markdown backticks.

```
{
  "g": "...",
  "P_t": ["subtask 1", "subtask 2"],
  "R_t": ["subtask 3"],
  "e_t": "...",
  "C": ["constraint 1"],
  "F_t": [{"action": "...", "step": 8, "cause": "..."}],
  "K_t": ["fact 1", "fact 2"]
}
```

You have access to the ground-truth environment state that the agent did NOT have. Use this to identify where the agent's observations were stale or its causal attributions were wrong.

## I Use of AI Assistants

AI assistants were used to help revise, brainstorm section organization, improve clarity, and debug. All experimental results, code, data analysis, and numerical claims were produced and verified by the authors. The authors reviewed and approved all text in the final submission.

## J Licenses

### J.1 Benchmarks and Environments

- **WebArena:** Apache-2.0 License
- **Terminal-Bench:** Apache-2.0 License

### J.2 Models

- **Qwen3.5-27B:** Apache-2.0 License
- **GPT-5.2:** Proprietary; no open-source license. Accessed via the OpenAI API.
- **Claude Sonnet 4.6:** Proprietary; no open-source license. Accessed via the Anthropic API.

### **J.3 Agent Architectures and Baselines**

- **ReAct**: No explicit license in the reference repository. Reimplemented from scratch in our harness.
- **StateAct**: MIT License. Reimplemented from scratch in our harness.
- **ACON**: CC BY-SA 4.0. Reimplemented from scratch.
- **AgentDiet**: MIT License. Reimplemented from scratch.
- **Sliding Window**: Reimplemented from scratch.

All implementations will be released under the MIT License upon publication.