When Harnesses Lose the Signal: Causal Evaluation of Recovery in LLM Agents

Anonymous authors
Paper under double-blind review

Abstract

Large language model agents rely on external harnesses to pass information between the model and its environment and to recover from execution errors. Yet recovery is usually judged only by average task success. This hides an important tension. The same operation can rescue a failing trajectory or disrupt one that would otherwise succeed. We frame recovery as a causal decision problem. Starting from the same execution state, we compare what happens with and without recovery, separate rescue from harm, and study how the value of recovery changes over time. We then introduce the Causal Intervention Router (CIR), a lightweight policy that uses information available before recovery to decide when intervention is worthwhile. On long-horizon ALFWorld tasks with Qwen3-14B, CIR raises success from 70.33% to 73.33%, a gain of 3.00 percentage points. It leaves all evaluated trajectories with correct observations untouched. Additional controls show that the benefit of recovery cannot be explained solely by the new observation returned by the environment. These results provide a practical way to evaluate recovery and apply it selectively. Code and artifacts are available at https://anonymous.4open.science/r/Causal-Evaluation-of-Recovery-in-LLM-Agents-308D/.

1 Introduction

Large language model agents do not act alone. An external harness passes observations between the model and its environment, maintains context, and coordinates long sequences of actions (Packer et al., 2023; Wu et al., 2024a; Yang et al., 2024; Yao et al., 2026). If an observation is outdated or missing, the agent may act on the wrong information, and the error can spread through later steps (Yao et al., 2023; Shah, 2026; Ma et al., 2026). A harness can respond by querying the environment again, which is called *refresh*.

Refresh is not always helpful. It may turn a failure into a success, but it may also derail a trajectory that would otherwise finish the task. We call these outcomes *rescue* and *harm*. Existing methods improve recovery through reflection, external feedback, or search (Shinn et al., 2023; Madaan et al., 2023; Gou et al., 2024; Zhou et al., 2024). Evaluations, however, usually report an overall success rate or identify where failures occur (Ma et al., 2024; Cemri et al., 2025; Zhang et al., 2025). These summaries do not show which trajectories recovery changes. The same average gain can hide very different numbers of rescues and harms.

This creates two challenges. First, an ordinary run shows only one outcome. The agent either recovers or continues without recovery (Holland, 1986). Without the alternative outcome, we cannot tell whether recovery caused the final result. Second, timing matters. After an error, later actions and feedback change the situation. An intervention that helps now may be ineffective or harmful later. A useful evaluation must compare both choices from the same state and repeat that comparison at different times.

Together, these challenges expose a gap between evaluating a recovery operation and deciding when to use it. A positive average effect does not imply that recovery is safe to apply broadly. The same average can result from a few reliable rescues or from many rescues offset by many harms. A modest average can also hide a large benefit in a recognizable subset of states. The central question is whether refresh works for a given trajectory and when its expected benefit outweighs its risk.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

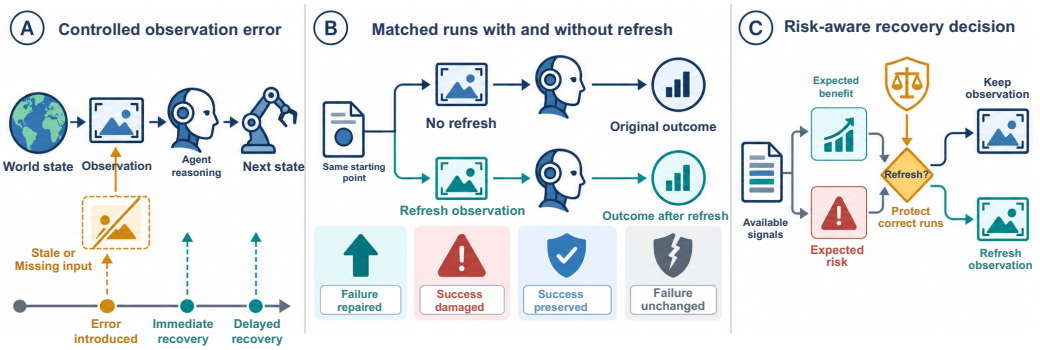


Figure 1: Overview of our framework. (A) We introduce controlled observation errors and try recovery at different times. (B) From the same execution point, paired runs with and without refresh reveal rescue, harm, and unchanged outcomes. (C) CIR uses information available before intervention to decide whether the expected benefit of refresh outweighs its risk.

Figure 1 shows our approach. We first introduce controlled observation errors. From the same execution point, we create two continuations, one with refresh and one without. This paired comparison reveals whether refresh rescues the task, harms it, or leaves the outcome unchanged. We also compare refresh variants to identify what drives the effect. We then use these paired outcomes to train the Causal Intervention Router (CIR). CIR decides whether to refresh using information available at decision time. It also limits harm to trajectories with correct observations.

The paired design serves two purposes. As an evaluation tool, it measures the effect of recovery while holding the earlier trajectory fixed. As a source of supervision, it reveals the states in which recovery changes the final outcome. The same paired outcomes used for evaluation also train CIR. At test time, it uses only information available in a live run.

Experiments on long-horizon ALFWorld tasks show that the value of refresh varies across errors, intervention times, and trajectories. They also show that a newly returned observation is not the sole source of the benefit. CIR turns this evidence into a selective policy. For Qwen3-14B, it improves success from 70.33% to 73.33% and does not refresh any evaluated clean trajectory. Our main contributions are as follows.

1. We introduce a paired framework for evaluating harness recovery. By continuing from the same pre-recovery state with and without an intervention, the framework attributes changes in the final outcome to recovery and separates rescue from harm. Comparisons across times and recovery variants show when refresh helps and which parts of the operation matter.
2. We introduce CIR, which uses the paired evidence to decide when recovery is worth the risk. It requires no retraining of the underlying agent and uses only information available before intervention. On Qwen3-14B, CIR improves overall success by 3.00 percentage points. The gain reaches 9.33 points when the agent receives a more severely outdated observation. All evaluated clean trajectories remain untouched.

2 Related Work

Agent Harnesses and Recovery Mechanisms. Agent harnesses organize interactions among models, tools, and environments, so their design can directly affect task performance (Yao et al., 2026). AutoGen (Wu et al., 2024a) provides composable conversation patterns, SWE-agent (Yang et al., 2024) studies agent and computer interfaces, and StateFlow (Wu et al., 2024b) uses explicit states to manage long tasks. Recovery methods revise execution through reflection, feedback, or search (Li et al., 2026). Reflexion (Shinn et al., 2023) learns from task feedback, Self-Refine (Madaan et al., 2023) revises outputs iteratively, CRITIC (Gou et al., 2024) verifies outputs with external tools, and LATS (Zhou et al., 2024) combines feedback with tree search. These methods improve how agents recover, but their aggregate evaluations do not identify the trajectories changed by recovery or the successful trajectories that recovery disrupts. This gap matters because

self-correction without reliable feedback can degrade an initially correct response (Huang et al., 2024). We instead measure rescue and harm from the same starting state and use those outcomes to learn when recovery should be applied.

Agent Failure Diagnosis and Causal Intervention. Process-level evaluation studies where and why agent failures occur (Barke et al., 2026). AgentBoard (Ma et al., 2024) tracks fine-grained progress, MAST (Cemri et al., 2025) provides a failure taxonomy for multi-agent systems, Who&When (Zhang et al., 2025) identifies responsible agents and steps, and CatchBench (Zhao et al., 2026) audits failures from configurations, live prefixes, and completed traces. Counterfactual approaches move from describing failures toward repairing them. Causal Agent Replay (Shah, 2026) estimates the contribution of individual steps, CausalFlow (Bonagiri et al., 2026) generates targeted repairs, HarnessFix (Chen et al., 2026) diagnoses and patches harness flaws, and DoVer (Ma et al., 2025) tests debugging hypotheses through intervention. These methods localize causes or repair observed failures. They do not directly answer our question. From the same current state, will a given recovery operation improve or damage the final outcome? When should it be applied? We answer these questions with paired runs and a policy that limits observed harm on clean trajectories.

This distinction separates our work from both failure localization and repair generation. We keep the recovery operation fixed and study its causal value across states and times. This makes rescue and harm directly comparable and allows the same evidence used for evaluation to train a deployment-time decision rule.

3 Paired Counterfactual Evaluation of Recovery

An aggregate success rate cannot tell us whether recovery changes a particular trajectory. We therefore compare recovery with continued execution from the same task state. Controlled changes to the observation make the error explicit, while paired runs isolate the recovery decision and reveal both rescue and harm (Figure 1(A–B)).

3.1 Perturbation and Recovery Setting

The unit of evaluation is a task instance, denoted by i. At a predefined point in the run, the harness keeps the environment unchanged and modifies only the observation shown to the agent. The observation condition e is clean, stale, or missing. A clean observation preserves the current information. A stale observation replaces it with information from an earlier state, and a missing observation removes its content. In this way, we change the information available to the agent without changing the underlying task.

The recovery delay d is the number of environment actions between the modified observation and the recovery operation m. A refresh queries the environment again and gives the agent the current observation. Replanning instead asks the agent to reconsider its next actions using information already in context. The triple (e, d, m) therefore specifies what information the agent receives, when recovery occurs, and which recovery operation is used. Section 5.1 gives the concrete settings.

3.2 Counterfactual Comparison

For each task i and observation condition e, execution continues until delay d. We then branch the run from exactly the same state. One branch continues without recovery. The other applies operation m. The branches share the environment state and the full execution history before this decision.

Because the two branches use the same state, history, model, and decoding settings, their difference is attributable to the recovery decision for the states studied here.

Using potential-outcome notation (Rubin, 1974), let $Y_i^0(e)$ and $Y_i^m(e, d)$ denote the binary task outcomes without recovery and with operation m at delay d, respectively. Under deterministic replay, the no-recovery continuation is shared across delay comparisons, so we suppress d in $Y_i^0(e)$. The instance-level effect of recovery is

$$\tau_i^m(e, d) = Y_i^m(e, d) - Y_i^0(e). \tag{1}$$

3.3 RESCUE AND HARM

We distinguish two directions of recovery effects. A *rescue* occurs when recovery turns a failure into a success, whereas *harm* occurs when recovery turns a success into a failure.

Let $\mathbf{1}[\cdot]$ denote the indicator function, which equals 1 when the enclosed condition holds and 0 otherwise. The rescue indicator $\mathcal{R}_i^m(e,d)$ and harm indicator $\mathcal{H}_i^m(e,d)$ for instance i are defined as

$$\mathcal{R}_i^m(e,d) = \mathbf{1}\left[Y_i^0(e) = 0,\ Y_i^m(e,d) = 1\right], \mathcal{H}_i^m(e,d) = \mathbf{1}\left[Y_i^0(e) = 1,\ Y_i^m(e,d) = 0\right]. \quad (2)$$

If both continuations succeed, recovery preserves success. If both fail, recovery leaves the failure unchanged. Together with rescue and harm, these cases form the four possible outcomes of a paired comparison. The instance-level recovery effect can therefore be written as

$$\tau_i^m(e,d) = \mathcal{R}_i^m(e,d) - \mathcal{H}_i^m(e,d). \quad (3)$$

The average recovery effect equals the rescue rate minus the harm rate. The same net effect can arise from rare changes or from frequent changes in both directions. The average alone cannot distinguish these cases. The paired outcomes also provide supervision for learning when recovery is more likely to rescue than harm a trajectory.

This view changes what counts as a successful recovery method. A method should not be judged only by how often it succeeds after intervention, because many of those trajectories may already have succeeded without intervention. Instead, evaluation should ask whether the method changes the outcome, in which direction, and at what point in the trajectory. The paired design answers all three questions with the same set of outcomes.

4 FROM EVALUATION TO SELECTIVE RECOVERY

Paired evaluation tells us whether recovery helped only after both branches have finished. In a live run, the harness must decide before either outcome is known. CIR estimates the value of recovery from the current state. It refreshes only when the expected benefit justifies the risk. A refresh obtains a new observation from the environment and adds it to the context. CIR refreshes at most once per trajectory (Figure 1(C)).

CIR makes a simple distinction. Detecting that something looks wrong is different from predicting that a particular response will help. An anomaly detector addresses the first question. A recovery policy must also answer the second. Some erroneous trajectories are not rescued by refresh, and some apparently normal trajectories can be harmed by it. CIR combines error detection with separate predictions of task success under refresh and continued execution.

4.1 DECISION OBJECTIVE

Let $X_{i,d}$ denote the information available before recovery at delay d. The decision $\pi(X_{i,d}) \in \{0,1\}$ indicates whether to refresh or continue for now. Let $Y_i^\pi(e)$ denote the final binary outcome under this sequential policy. If no refresh is triggered, $Y_i^\pi(e) = Y_i^0(e)$. Otherwise, it equals the outcome after refresh at the policy-selected delay.

Let $\epsilon \in [0,1]$ denote the allowed harm rate on clean trajectories, and let $\mathbb{E}[\cdot]$ denote expectation over task instances under the specified observation condition. We denote an optimal policy by $\pi^\star$ and formulate selective recovery as

$$\begin{aligned} \pi^\star &= \arg\max_\pi \mathbb{E}\left[Y_i^\pi(e) - Y_i^0(e) \mid e \neq \text{clean}\right], \\ \text{subject to} \quad & \mathbb{E}\left[Y_i^0(e)\left(1 - Y_i^\pi(e)\right) \mid e = \text{clean}\right] \leq \epsilon. \end{aligned} \quad (4)$$

The objective rewards improvement when the observation is wrong, relative to never recovering. Maximizing gain alone could encourage frequent refreshes that disrupt correct executions. The second line limits this risk. Its value is the fraction of clean trajectories that succeed without recovery but fail under the policy. At runtime, neither possible outcome is known. CIR must therefore estimate the value of refresh from information available before the decision.

4.2 Estimating Recovery Value

Detecting a suspicious observation is not enough. An error may exist even when refresh would not improve the final outcome. CIR separates two questions. Did the agent receive a faulty observation? How likely is the task to succeed with or without refresh? It answers them using execution behavior, consistency between actions and observations, task progress, and contextual information.

Let x denote a value of $X_{i,d}$, and let $\Pr(\cdot \mid \cdot)$ denote conditional probability. We use $p_{\mathrm{err}}(x)$ for the probability that the trajectory has received an erroneous observation, $p_0(x)$ for the probability of success without subsequent recovery, and $p_1(x)$ for the probability of success after immediate refresh. Denoting refresh by ref and its final outcome at delay d by $Y_i^{\mathrm{ref}}(e, d)$, we define

$$\begin{aligned} p_{\mathrm{err}}(x) &= \Pr(e \neq \text{clean} \mid X_{i,d} = x)\,, \\ p_0(x) &= \Pr\big(Y_i^0(e) = 1 \mid X_{i,d} = x\big)\,, \\ p_1(x) &= \Pr\big(Y_i^{\mathrm{ref}}(e, d) = 1 \mid X_{i,d} = x\big)\,. \end{aligned} \tag{5}$$

CIR estimates these quantities with three models supervised by the observation condition, the no-recovery outcome, and the refresh outcome in the paired data from Section 3. The two outcome models follow a T-learner-style design and estimate the two decision outcomes separately (Künzel et al., 2019). This separation prevents high anomaly probability from being treated as evidence that refresh is beneficial. For brevity, we use the same notation for the probabilities and their estimates.

To account for potential rescue and harm, we construct a rescue score $s_{\mathrm{R}}(x)$ and a harm score $s_{\mathrm{H}}(x)$.

$$s_{\mathrm{R}}(x) = \big(1 - p_0(x)\big)p_1(x), \qquad s_{\mathrm{H}}(x) = p_0(x)\big(1 - p_1(x)\big). \tag{6}$$

The rescue score is high when continued execution is likely to fail and refresh is likely to succeed. The harm score is high when continued execution is likely to succeed and refresh is likely to fail. These scores help identify states in which refresh is more promising. The marginal success probabilities do not determine the probability that the same task is rescued or harmed (Tian & Pearl, 2000). We therefore do not interpret these scores as rescue or harm probabilities.

Let $\lambda > 0$ denote the harm penalty. Let $u_\lambda(x)$ denote the utility obtained by subtracting the weighted harm score from the rescue score.

$$u_\lambda(x) = s_{\mathrm{R}}(x) - \lambda s_{\mathrm{H}}(x). \tag{7}$$

When $\lambda = 1$, this expression reduces to $p_1(x) - p_0(x)$, the difference in predicted success probabilities between refresh and no recovery. Increasing λ lowers the utility of states with high harm scores, making refresh less likely to be triggered in those states. Thus, λ controls the weight assigned to harm in each decision, whereas ϵ limits the fraction of all clean trajectories that the policy turns from success into failure.

4.3 Dynamic Recovery Decisions

The value of recovery can change as the agent takes more actions and receives more feedback. CIR therefore reconsiders refresh at several candidate times instead of committing to one fixed delay. At each time, it compares an immediate refresh with continuing without later recovery. If it chooses to continue, it can reconsider after the state has changed.

Let $p_{\mathrm{fail}}(x) = 1 - p_0(x)$ denote the predicted probability of failure without subsequent recovery. Let α, β, and γ denote the thresholds for the error probability, failure probability, and utility score. Let $d_{\min}$ denote the earliest allowed refresh delay. The indicator function $\mathbf{1}[\cdot]$ was defined in Section 3. The decision rule before any refresh has occurred is

$$\pi(X_{i,d}) = \mathbf{1}\begin{bmatrix} d \geq d_{\min}, \\ p_{\mathrm{err}}(X_{i,d}) \geq \alpha, \\ p_{\mathrm{fail}}(X_{i,d}) \geq \beta, \\ u_\lambda(X_{i,d}) \geq \gamma \end{bmatrix}. \tag{8}$$

CIR checks candidate times in chronological order and refreshes at the first time when all conditions are satisfied. No further recovery is triggered afterward. If no candidate time satisfies the conditions, the task proceeds without recovery.

We select the decision thresholds on development data. Among the settings whose observed harm rate on clean trajectories does not exceed ϵ, we choose the one with the largest net gain under erroneous observations. We then apply that fixed policy to new tasks. This procedure is related to empirical welfare maximization (Kitagawa & Tetenov, 2018), with paired recovery outcomes defining the benefit and the clean-trajectory results defining the risk limit.

5 Experiments

Our experiments answer three questions. When does refresh help or harm? Which parts of refresh produce the benefit? Can CIR use this evidence to recover selectively on new tasks? We first describe the common setup and then answer each question.

5.1 Experimental Setup

Task, agent, and perturbations. We use ALFWorld (Shridhar et al., 2021), where an agent completes long household tasks through text actions and observations. Our agent is Qwen3-14B (Yang et al., 2025), used with a fixed ReAct-style harness (Yao et al., 2023) and greedy decoding. Each run is limited to 49 model turns and 64 environment actions. We generated the agent trajectories on eight NVIDIA H200 GPUs. We first run each task without an injected error or recovery. We call this the *factual trajectory*. The observation is modified after 25% of the environment actions in that trajectory.

The primary perturbation is a two-step stale observation, which replaces the current observation with the observation from two environment actions earlier. Additional analyses use a one-step stale observation from one environment action earlier and a missing observation whose content is replaced by a fixed unavailable-observation message. A refresh executes an additional `look` action and appends the returned current observation to the transcript. We test recovery delays $d \in \{0, 1, 2, 4\}$.

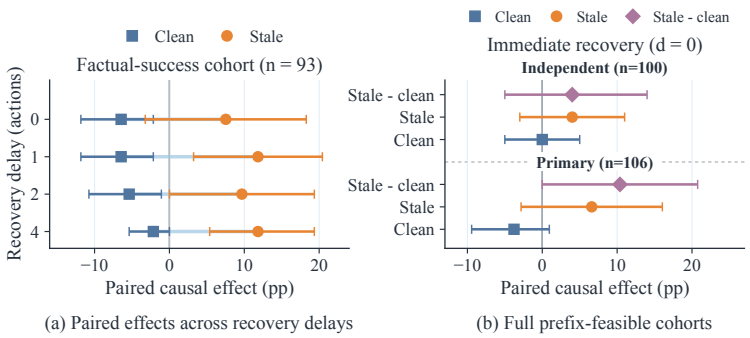
Paired runs and data partitions. Each comparison starts two runs from the same environment state and model input, with identical model and decoding settings. The runs differ only in whether the assigned recovery operation is applied. A task is *prefix-feasible* if it reaches the chosen intervention point and contains enough earlier history to construct the required stale observation. We also verify that the two runs have identical histories before they branch.

Each cohort serves a distinct role. The primary and independent cohorts test whether recovery effects replicate. The mechanism cohort isolates the components of refresh. The evaluation cohort tests CIR. Some analyses focus on tasks whose factual trajectory succeeds. To check that the conclusions do not depend on this filter, we also report complete cohorts that retain every prefix-feasible task. This gives 106 tasks in the primary cohort and 100 separately executed tasks in the independent cohort. These cohorts cover the available prefix-feasible tasks, not the full ALFWorld distribution. The labels *factual success* and *factual failure* describe the unmodified run. Rescue and harm always compare recovery with no recovery under the same observation condition.

The primary factual-success analysis retains 93 of the 106 `valid_unseen` tasks, while the independent cohort contributes 89 factual-success tasks. Mechanism controls use a separate 74-task set, of which 67 pass the paired-analysis protocol checks detailed in Appendix A.

CIR is fitted on the 93 primary tasks and evaluated on a separate set of 76 `valid_seen` tasks. The names `valid_seen` and `valid_unseen` refer to ALFWorld splits. No task appears in both fitting and evaluation. Of the evaluation tasks, 75 are prefix-feasible. Each is tested under four observation conditions, giving 300 evaluation episodes. We apply the fitted models and decision thresholds without further tuning. All tasks remain in the denominator at every delay. If a task has already ended, we carry its final outcome forward and do not apply recovery.

Estimation and metrics. Our main quantity is the paired difference in task success between recovery and no recovery, measured in percentage points (pp). At delay d, we also compare this effect between two-step stale and clean observations. We report the paired outcome counts from Section 3.3. Confidence intervals use a task-level cluster bootstrap. This keeps all conditions from the same task together in each resample (Efron, 1979). For learned policies, we additionally report how often the policy intervenes and how often it harms a clean trajectory.



(a) Paired effects across recovery delays (b) Full prefix-feasible cohorts

Figure 2: Paired recovery effects in the 93-task factual-success cohort across delays (a) and in the complete prefix-feasible primary ($n = 106$) and independent ($n = 100$) cohorts (b). "Stale" denotes the two-step stale condition. Error bars are task-level cluster-bootstrap 95% confidence intervals.

CIR fitting and comparison policies. CIR fits three ℓ_2-regularized logistic models with $C = 0.1$ to estimate the observation-error probability and the two outcome probabilities in Eq. equation 5. Inputs contain only information available before a candidate intervention, including action and reasoning repetition, no-op behavior, observation novelty, consistency between actions and observations, task progress, and context-budget statistics.

To keep model fitting separate from threshold selection, we divide the 93 training tasks into five groups. In turn, we fit the models on four groups and predict the held-out group. This gives every training task an out-of-fold (OOF) prediction from models that did not see that task. We choose the decision thresholds from these predictions and then refit the models on all training tasks. All model parameters and thresholds are fixed before test evaluation. We use $\lambda = 2$, $\epsilon = 0.02$, $d_{\min} = 1$, and thresholds $\alpha = 0.876$, $\beta = 0.052$, and $\gamma = 0$. During evaluation, CIR checks $d \in \{1, 2, 4\}$ in order and refreshes at most once.

We compare CIR with never refreshing and with a sequential anomaly baseline. The anomaly baseline uses the same error detector at each candidate time but does not predict whether refresh will help. Its threshold is selected from the same OOF predictions, objective, and clean-harm limit used for CIR. We also include a version whose training-time intervention rate is matched to CIR. Neither baseline is adjusted on the test set.

5.2 Recovery Depends on State and Timing

We first ask whether the same refresh operation has different effects in different states and at different times. We begin with the 93 primary tasks whose factual trajectory succeeds. Under clean observations, continued execution reproduces a known success, so any failure after refresh is directly interpretable as harm. Figure 2(a) shows this cohort. An immediate refresh lowers success by 6.45 percentage points under clean observations (95% CI $[-11.83, -2.15]$). It raises success by 7.53 points under two-step stale observations (95% CI $[-3.23, 18.28]$). The difference between these effects is 13.98 points (95% CI $[2.15, 24.73]$). The stale-minus-clean difference remains positive at later delays. The estimates are 18.28, 15.05, and 13.98 points at delays 1, 2, and 4.

In this cohort, continuing without recovery under a clean observation reproduces the successful factual trajectory, so $Y_i^0(\text{clean}) = 1$ by construction. Refresh can preserve or harm that success, but it cannot create a rescue. We therefore repeat the analysis with factual failures included.

The independent 89-task factual-success cohort shows the same ordering at every tested delay. The effect under stale observations is more positive than the effect under clean observations, with differences ranging from 8.99 to 14.61 points. At delay 2, refresh changes success by $+10.11$ points under stale observations and -4.49 points under clean observations.

We next include every prefix-feasible task in both cohorts. This includes 13 factual failures in the primary cohort and 11 in the independent cohort. Figure 2(b) summarizes the results. In the 106-task primary cohort, refresh changes success by -3.77 points for clean observations (95% CI $[-9.43, 0.94]$) and $+6.60$ points for stale observations (95% CI $[-2.83, 16.04]$). The difference is

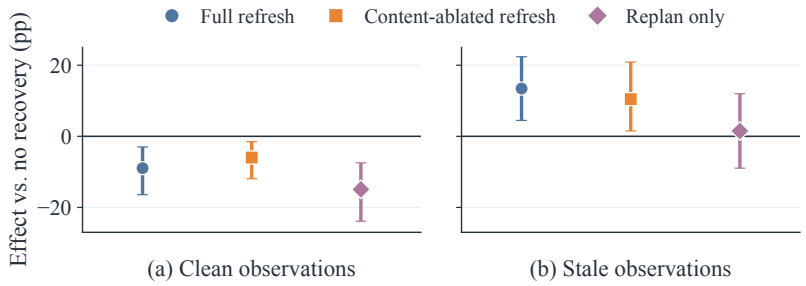


Figure 3: Mechanism controls under clean observations at delay 0 (a) and two-step stale observations at delay 4 (b). Points are effects relative to no recovery. Error bars are task-level cluster-bootstrap 95% confidence intervals.

10.38 points (95% CI $[0.00, 20.75]$). In the 100-task independent cohort, the corresponding effects are 0.00 and $+4.00$ points. Their difference is 4.00 points (95% CI $[-5.00, 14.00]$). The estimated effect is therefore more favorable under stale observations in both complete cohorts, including tasks with failed factual trajectories.

The paired counts show what these averages hide. In the complete primary cohort, refresh produces 2 rescues and 6 harms under clean observations, compared with 17 rescues and 10 harms under stale observations. At delay 1 in the factual-success cohort, the positive average under stale observations combines 15 rescues with 4 harms. Under clean observations, refresh turns 6 successes into failures. Appendix B gives the full breakdown.

Taken together, these results show that recovery value is not a fixed property of the refresh operation. It depends on the information available to the agent, the state reached after the error, and the time at which recovery is attempted. This explains why a single unconditional rule can hide useful interventions and unnecessary ones behind the same average success rate.

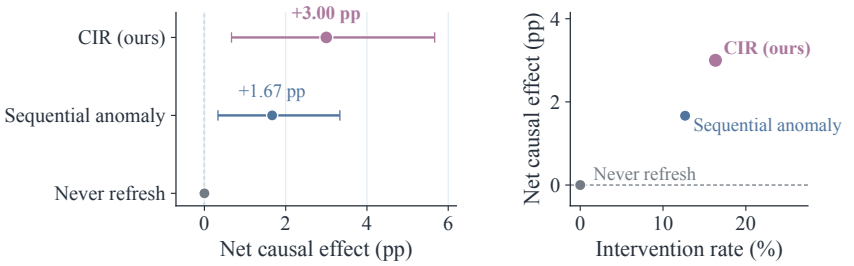
5.3 Why Does Refresh Help?

We next ask what part of refresh produces the gain. A full refresh queries the environment and shows the returned observation to the agent. A second variant, called content-ablated refresh, makes the same query but hides the returned content. A third variant asks the agent to replan without querying the environment. Under clean observations at delay 0, these operations change success by -8.96, -5.97, and -14.93 percentage points, respectively (Figure 3). Under two-step stale observations at delay 4, the changes are $+13.43$ points (95% CI $[4.48, 22.39]$), $+10.45$ points (95% CI $[1.49, 20.90]$), and $+1.49$ points (95% CI $[-8.96, 11.98]$). Full refresh rescues ten tasks. Content-ablated refresh also rescues eight of them. The returned observation is therefore not required for those shared rescues. Appendix A gives the task-level comparison. This result matters because refresh is often treated as if it were equivalent to supplying better observation content. Our controls show that refresh has two distinct components. It queries the environment and then shows the returned content to the agent. The returned content is unnecessary for most shared rescues. Evaluating only the full operation would miss this distinction.

5.4 CIR Learns When to Recover

We evaluate CIR on all 75 held-out prefix-feasible tasks, whether or not their factual trajectory succeeds. Across 300 episodes, never refreshing succeeds on 70.33%, while CIR reaches 73.33%. The gain is 3.00 percentage points (95% CI $[0.67, 5.67]$), with 11 rescues and 2 harms.

The effect varies by observation condition (Figure 4 and Table 1). CIR never intervenes when the observation is clean, so success is unchanged. Its largest gain occurs when the observation is two steps out of date. Success rises from 60.00% to 69.33%, an improvement of 9.33 points (95% CI $[1.33, 17.33]$) from 9 rescues and 2 harms. The gains for one-step stale and missing observations are each 1.33 points. CIR intervenes in 31 of the 75 two-step stale episodes and in only 1 of the 75 missing-observation episodes. It concentrates recovery where the observed benefit is largest.



(a) Net effect on task success (b) Effect–intervention trade-off

Figure 4: Selective recovery on 75 held-out prefix-feasible tasks. Each task is evaluated under four observation conditions, giving 300 episodes. Points show effects relative to never refreshing. Error bars are task-level cluster-bootstrap 95% confidence intervals. CIR intervenes in 16.3% of episodes.

Observation condition	Never	CIR	Effect (pp, 95% CI)	Rescue / harm
Clean	81.33%	81.33%	0.00 [0.00, 0.00]	0 / 0
One-step stale	68.00%	69.33%	+1.33 [0.00, 4.00]	1 / 0
Two-step stale	60.00%	69.33%	+9.33 [1.33, 17.33]	9 / 2
Missing	72.00%	73.33%	+1.33 [0.00, 4.00]	1 / 0

Table 1: Test performance of CIR by observation condition on all 75 held-out prefix-feasible tasks.

CIR intervenes in 16.3% of episodes. The sequential anomaly baseline intervenes in 12.7% and improves success by 1.67 points (95% CI $[0.33, 3.33]$), with 6 rescues and 1 harm. The rate-matched anomaly baseline intervenes in 15.7% and improves success by 2.00 points (95% CI $[0.67, 3.67]$), with 6 rescues and no harms. CIR has the highest point estimate. Its margins over the two anomaly baselines are 1.33 points (95% CI $[-0.33, 3.33]$) and 1.00 point (95% CI $[-0.67, 2.67]$), respectively. The 3.00-point gain over never refreshing is supported by a task-level sign-flip test ($p = 0.034$). The rate-matched comparison helps separate selective recovery from simply intervening more often. CIR and the rate-matched anomaly policy refresh a similar fraction of episodes. CIR produces 11 rescues rather than 6 and attains the highest observed success rate. This pattern is consistent with CIR's design. The policy considers the predicted outcome of the intervention, not only whether the current observation appears unusual. CIR does not trigger on any of the 75 clean episodes, preserving the never-refresh success rate throughout the evaluated clean cohort.

Discussion. Recovery should be evaluated as a decision, not only as an operation. Paired continuations reveal which trajectories recovery rescues or harms and how its value changes with state and timing. They provide more information than post-recovery success alone. They also show why anomaly detection is insufficient. An unusual observation does not imply that refresh will help. CIR predicts outcomes under refresh and continued execution before acting. A harness can therefore estimate recovery value first and intervene only when the expected value is positive.

6 Conclusion

Average success alone hides how recovery changes individual trajectories. Our paired evaluation shows that the value of refresh depends on the observation, the timing, and the trajectory. It also separates the tasks that recovery rescues from those it harms. Controlled refresh variants show that the newly returned observation is not the only source of benefit. CIR turns this evidence into a practical decision. Without retraining the underlying agent, it improves success by 3.00 percentage points across 300 episodes from 75 held-out tasks. Its largest gain occurs when observations are more severely outdated, and it attains the highest observed success rate among the policies we compare. These results show how paired causal evidence can support more selective recovery in the harness of LLM agents.

AI USE STATEMENT

In this work, we used generative AI tools only for language polishing, including improving grammar, clarity, and fluency of the manuscript. We have not used generative AI tools for generating research ideas, developing the methodology, designing experiments, analyzing results, or drawing scientific conclusions; other disclosure categories are not applicable to this work. We reviewed all AI-assisted edits and verified that they did not alter the technical content or scientific claims. We take full responsibility for the final content of this work, including all text, claims, and artifacts.

ETHICS STATEMENT

All experiments are conducted in controlled benchmark environments and do not involve human subjects or private user data. We do not identify additional ethical risks beyond those commonly associated with LLM agent research.

REPRODUCIBILITY STATEMENT

The paper specifies the main causal variables, intervention definitions, evaluation metrics, models, benchmarks, and experimental protocols. Code, experimental artifacts, and additional implementation details are available at https://anonymous.4open.science/r/Causal-Evaluation-of-Recovery-in-LLM-Agents-308D/.

REFERENCES

Shraddha Barke, Arnav Goyal, Alind Khare, Avaljot Singh, Suman Nath, and Chetan Bansal. Agentrx: Diagnosing ai agent failures from execution trajectories, 2026. URL https://arxiv.org/abs/2602.02475.

Akash Bonagiri, Devang Borkar, Gerard Janno Anderias, Setareh Rafatirad, and Houman Homayoun. CausalFlow: Causal attribution and counterfactual repair for LLM agent failures, 2026. URL https://arxiv.org/abs/2605.25338. arXiv preprint arXiv:2605.25338.

Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei A. Zaharia, Joseph E. Gonzalez, and Ion Stoica. Why do multi-agent LLM systems fail? In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-4082. URL https://papers.nips.cc/paper_files/paper/2025/hash/b1041e52d3be19f0a9bc491657488e4a-Abstract-Datasets_and_Benchmarks_Track.html.

Mengzhuo Chen, Junjie Wang, Zhe Liu, Yawen Wang, Haiming Zheng, and Qing Wang. From failed trajectories to reliable LLM agents: Diagnosing and repairing harness flaws, 2026. URL https://arxiv.org/abs/2606.06324. arXiv preprint arXiv:2606.06324.

Bradley Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7 (1):1–26, 1979. doi: 10.1214/aos/1176344552. URL https://doi.org/10.1214/aos/1176344552.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2305.11738.

Paul W. Holland. Statistics and causal inference. *Journal of the American Statistical Association*, 81 (396):945–960, 1986. doi: 10.1080/01621459.1986.10478354. URL https://www.jstor.org/stable/2289064.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.01798.

Toru Kitagawa and Aleksey Tetenov. Who should be treated? empirical welfare maximization methods for treatment choice. *Econometrica*, 86(2):591–616, 2018. doi: 10.3982/ECTA13288. URL https://onlinelibrary.wiley.com/doi/10.3982/ECTA13288.

Sören R. Künzel, Jasjeet S. Sekhon, Peter J. Bickel, and Bin Yu. Metalearners for estimating heterogeneous treatment effects using machine learning. *Proceedings of the National Academy of Sciences*, 116(10):4156–4165, 2019. doi: 10.1073/pnas.1804597116. URL https://www.pnas.org/doi/10.1073/pnas.1804597116.

Shiyu Li, Yifan Wang, Peiming Li, Zheng Wei, and Yang Tang. Reseek: A self-correcting framework for search agents with instructive rewards. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. AgentBoard: An analytical evaluation board of multi-turn LLM agents. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/877b40688e330a0e2a3fc24084208dfa-Abstract-Datasets_and_Benchmarks_Track.html.

Ming Ma, Jue Zhang, Fangkai Yang, Yu Kang, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. DoVer: Intervention-driven auto debugging for LLM multi-agent systems, 2025. URL https://arxiv.org/abs/2512.06749. arXiv preprint arXiv:2512.06749.

Ziyu Ma, Hailang Huang, Shun Zou, Yong Wang, Shidong Yang, Yiming Hu, Fei Wei, and XiangXiang Chu. Longhorizon-harness: Advancing long-horizon agents for real-world tasks, 2026. URL https://arxiv.org/abs/2608.01964.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems, 2023. URL https://arxiv.org/abs/2310.08560. arXiv preprint arXiv:2310.08560.

Donald B. Rubin. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of Educational Psychology*, 66(5):688–701, 1974. doi: 10.1037/h0037350. URL https://doi.org/10.1037/h0037350.

Jaineet Shah. Causal Agent Replay: Counterfactual attribution for LLM-agent failures, 2026. URL https://arxiv.org/abs/2606.08275. arXiv preprint arXiv:2606.08275.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.03768.

Jin Tian and Judea Pearl. Probabilities of causation: Bounds and identification. *Annals of Mathematics and Artificial Intelligence*, 28:287–313, 2000. doi: 10.1023/A:1018912507879. URL https://link.springer.com/article/10.1023/A:1018912507879.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024a. URL https://openreview.net/forum?id=BAakY1hNKS.

Yiran Wu, Tianwei Yue, Shaokun Zhang, Chi Wang, and Qingyun Wu. StateFlow: Enhancing LLM task-solving through state-driven workflows. In *First Conference on Language Modeling*, 2024b. URL https://arxiv.org/abs/2403.11322.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388. arXiv preprint arXiv:2505.09388.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5a7c947568c1b1328ccc5230172e1e7c-Abstract-Conference.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629.

Yilun Yao, Xinyu Tan, Chao-Hsuan Liu, Yaoming Li, Zhengyang Wang, Wenhan Yu, Zhewen Tan, Yuxuan Tian, Guangxiang Zhao, Lin Sun, Xiangzheng Zhang, and Tong Yang. Harness-bench: Measuring harness effects across models in realistic agent workflows, 2026. URL https://arxiv.org/abs/2605.27922.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. Which agent causes task failures and when? on automated failure attribution of LLM multi-agent systems. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 76583–76599. PMLR, 2025. URL https://proceedings.mlr.press/v267/zhang25cq.html.

Yue Zhao, Mengyuan Li, Ruolin Li, Prince Zizhuang Wang, Shuli Jiang, Linsey Pang, Xiongye Xiao, and Xiyang Hu. CatchBench: When can an agent failure be caught?, 2026. URL https://arxiv.org/abs/2608.22808. arXiv preprint arXiv:2608.22808.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62138–62160. PMLR, 2024. URL https://proceedings.mlr.press/v235/zhou24r.html.



Figure 5: Joint recovery outcomes under two-step stale observations at delay $d = 4$. Rows denote full refresh and columns denote content-ablated refresh, each classified relative to the same no-recovery outcome. The analysis includes 65 tasks with matching execution prefixes and excludes two tasks that terminate successfully before recovery. Eight rescues are shared, two occur only under full refresh, and one occurs only under content-ablated refresh.

Table 2: Paired success effects on the 65 tasks where both recovery operations can be executed. Intervals use 10,000 task-level cluster-bootstrap resamples. All values are in percentage points.

Paired comparison	Effect (pp)	95% CI
Full refresh − no recovery	+13.85	[4.62, 23.08]
Content-ablated refresh − no recovery	+10.77	[1.54, 20.00]
Full refresh − content-ablated refresh	+3.08	[−4.62, 10.77]

A Task-Level Overlap between Recovery Operations

We analyze the joint outcomes of full refresh and content-ablated refresh under two-step stale observations at delay $d = 4$. Both outcomes are classified relative to the same no-recovery outcome. The mechanism-control records contain 74 tasks, and 67 satisfy the protocol checks. Two of these tasks terminate successfully before the scheduled recovery, so neither refresh operation is executed. They remain in the 67-task aggregate analysis in Section 5.3, but they are excluded from the joint analysis of executed interventions. Figure 5 shows the joint task-level outcomes. Table 2 reports the paired effects. The remaining 65 tasks share the same task identity and exact token prefix before recovery in both branches. No environment-replay mismatches were recorded.

The outcome categories agree on 59 of 65 tasks. These include eight shared rescues, 38 stable successes, and 13 stable failures. The six disagreements include two full-only rescues, one content-ablated-only rescue, two harms avoided by full refresh, and one harm introduced by full refresh relative to content-ablated refresh. The high overall agreement therefore includes 51 tasks whose outcomes neither operation changes. The eight shared rescues show that the returned content is not necessary for these observed benefits.

The paired net gains are nine tasks for full refresh versus no recovery, seven for content-ablated refresh versus no recovery, and two for full refresh versus content-ablated refresh. Removing the two already-terminated tasks changes the denominator from 67 to 65 but does not change these gains.

This explains the difference from the aggregate effects in Section 5.3. Both refresh variants produce positive gains relative to no recovery. Full refresh has a 3.08-pp point estimate over content-ablated refresh.

B Additional Recovery Results

Figure 6 gives the complete paired-outcome decomposition for the 93-task factual-success cohort at recovery delay $d = 1$. Under stale observations, refresh produces 15 rescues and 4 harms, alongside 59 stable successes and 15 stable failures. Under clean observations, 87 trajectories remain successful and 6 are harmed. This decomposition illustrates how the positive stale-state average arises from trajectory-level changes that aggregate success rates alone conceal.

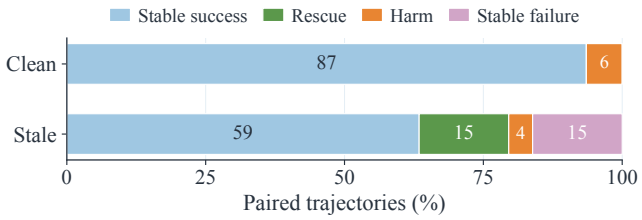


Figure 6: Paired outcomes at recovery delay $d = 1$ in the 93-task factual-success cohort. Each bar partitions trajectories into stable success, rescue, harm, and stable failure.