BENCH2ROBUST: SCENARIO-CONTROLLED RECOVERY FOR TOOL-USING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tool-using LLM agents are typically developed and post-trained in controlled environments with reliable tools, but deployment exposes them to tool failures, API and schema drift, and third-party services outside the developer's control. We introduce BENCH2ROBUST, a framework for scenario-controlled solvability that augments existing benchmarks with episodes requiring retrying the original path, switching to a fallback-equivalent tool, or recognizing that no viable path remains. By controlling recoverability, BENCH2ROBUST separates recovery-policy quality from randomness in whether failures persist or resolve. Across 7 models from 4 families and 10 task/domain slices spanning customer-service and function-calling benchmarks, 69 of 70 model-subset pairs degrade under tool failures, by up to 46.7 percentage points. We propose Bayesian Tool Memory (BTM) for structured runtime recovery context and use reinforcement learning (RL) for learned recovery behavior. On held-out Retail tasks using the benchmark's original tool set, BTM improves robustness by 16.8 percentage points without retraining, while RL retains a 6.3-point gain without inference-time BTM. Combining RL with BTM yields a 20.7-point improvement over the base model while preserving clean performance. These results show that robust tool use benefits from combining structured runtime recovery context with learned recovery behavior.

1 INTRODUCTION

Large language models (LLMs) augmented with tools can complete complex multi-turn tasks by retrieving information and acting through external systems (Schick et al., 2023; Qin et al., 2024; Patil et al., 2024). Most tool-use benchmarks, however, assume that tool calls return timely, structurally valid, correct, and current responses (Yao et al., 2025; Li et al., 2023). Real tools violate these assumptions: requests time out or hit rate limits, credentials and schemas become invalid, and apparently valid responses may contain stale, partial, or incorrect data. Moreover, tool failures vary in both persistence and observability: some resolve on retry, some persist, and others provide no explicit failure signal. Robust agents must therefore reason about what remains recoverable under changing tool conditions and choose an appropriate response: RETRY the current path, SWITCH to an alternative, or ABSTAIN and escalate once viable paths are exhausted (Figure 1).

A central challenge is that conventional stochastic failure injection does not specify what recovery behavior an episode actually requires. If a transient failure happens to disappear on the next call, indiscriminate retrying may appear robust. Conversely, a reasonable recovery action may still receive low reward if the sampled failure persists. As a result, observed task success can conflate recovery-policy quality with stochastic variation in the environment, making it difficult to ask a more fundamental question: *does the agent choose a recovery strategy that matches what remains solvable?*

We introduce BENCH2ROBUST, a benchmark-agnostic framework built around **scenario-controlled solvability**. Alongside stochastic tool-response corruption, BENCH2ROBUST assigns each episode to one of three controlled recoverability scenarios: in S1, the original path remains viable and retrying can recover; in S2, the primary path is blocked episode-wide and completion requires a fallback-equivalent tool; and in S3, all viable paths are blocked, making escalation the appropriate response. By fixing the recoverability scenario at the environment level, BENCH2ROBUST separates whether a task remains solvable from how the agent responds, establishing a controlled

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.

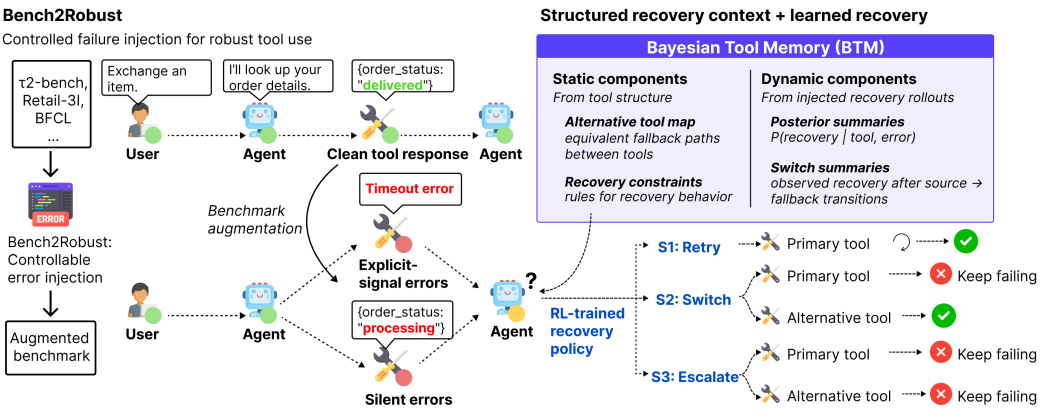

Figure 1: **Overview of BENCH2ROBUST and our recovery approach.** BENCH2ROBUST augments existing tool-use benchmarks with controllable tool-response failures, including explicit-signal errors and silent corruptions. Bayesian Tool Memory (BTM) combines static recovery structure—an alternative-tool map and recovery constraints—with dynamic recovery statistics estimated from injected recovery rollouts.

setting for both evaluation and recovery-oriented training. All three scenarios are used during training. Our held-out task-success evaluation focuses on solvable settings because existing benchmark evaluators do not reward correct stopping on intentionally unsolvable S3 episodes, and we avoid adding an explicit abstention reward that could incentivize premature stopping.

We use BENCH2ROBUST to study two complementary approaches to recovery. *Bayesian Tool Memory* (BTM) provides structured runtime context combining fallback relations, recovery constraints, and empirical recoverability statistics, while reinforcement learning trains recovery behavior across controlled scenarios. Our primary policy is Qwen3-4B, with a Qwen3-8B replication (Appendix D.1); BTM is used during RL exploration and optionally at inference.

We evaluate robustness across τ^2-bench (Barres et al., 2025), the Retail-3I and Airline-3I that extend the corresponding τ^2-bench domains (Wang et al., 2026a), and BFCL multi-turn (Patil et al., 2025), covering seven models from four model families. Recovery interventions are trained on Combined Retail (1,339 tasks from τ^2-bench Retail and Retail-3I) and evaluated on a 402-task held-out split, with Airline-3I, Telecom, and BFCL used for cross-benchmark evaluation.

Our contributions are threefold. First, we formulate tool-failure robustness as recovery under *controlled solvability*, explicitly separating episodes where retrying remains viable, switching is required, or no viable path remains. Second, we instantiate this framework across multiple tool-use benchmark families and characterize a broad robustness gap across models, domains, and failure regimes. Third, we study how structured runtime recovery context and RL training contribute separately and jointly to robustness on held-out tasks.

Our main findings are:

- **Tool failures expose a broad robustness gap.** Across 7 models, 4 families, and 10 task/domain slices, 69 of 70 model-subset pairs degrade under injection, by up to 46.7pp.
- **Structured recovery context provides large zero-training gains.** Online-updated BTM improves the 4B base model by +16.8pp without alternatives and +11.7pp with alternatives; ablations separate complementary contributions from static recovery structure and online adaptation.
- **RL produces recovery gains that persist without inference-time BTM.** Removing BTM at inference, the trained policy still improves robustness by +6.3/+6.9pp over the base model.
- **Runtime context and learned behavior contribute differently across error types.** BTM provides the largest gains on several explicit service/response errors, while the additional RL-associated gain is larger on auth/schema errors and factual corruption.

Algorithm 1 BENCH2ROBUST Scenario-Controlled Failure Injection

Require: Task τ, tool set $\mathcal{T}$, task-relevant fallback classes $\mathcal{E}_\tau$, viable tool paths $\mathcal{V}_\tau$, scenario $s \in \{\mathrm{S1}, \mathrm{S2}, \mathrm{S3}\}$, per-tool noise distributions $\{\boldsymbol{\theta}_j\}_{j \in \mathcal{T}}$, execution-suppressing/silent noise partition $\mathcal{N}_{\mathrm{exec}}, \mathcal{N}_{\mathrm{silent}}$, budget B, consecutive limit $K_{\max}$

$\mathcal{P} \leftarrow \emptyset$
if $s = \mathrm{S2}$ **then**
  Select a task-relevant equivalence class $E \in \mathcal{E}_\tau$
  Randomly choose one side $j \in E$ and set $\mathcal{P} \leftarrow \{j\}$
else if $s = \mathrm{S3}$ **then**
  $\mathcal{P} \leftarrow \mathcal{V}_\tau$
end if
for each tool call (j, a) in the episode **do**
  if $j \in \mathcal{P}$ **then**
    Return episode-persistent blocked response $\{(j, a)$ not executed; state unchanged$\}$
  else if stochastic injection budget remains and consecutive failures for $j < K_{\max}$ **then**
    Sample $\nu \sim \mathrm{Cat}(\boldsymbol{\theta}_j)$
    if $\nu \in \mathcal{N}_{\mathrm{exec}}$ **then**
      Return failure message for ν and update stochastic injection counters $\{(j, a)$ not executed; state unchanged$\}$
    else if $\nu \in \mathcal{N}_{\mathrm{silent}}$ **then**
      $o^\star \leftarrow Z_{\mathrm{clean}}(s_t, (j, a))$ $\{(j, a)$ executes normally$\}$
      Return $C_\nu(o^\star)$ and update stochastic injection counters
    end if
  end if
  $o^\star \leftarrow Z_{\mathrm{clean}}(s_t, (j, a))$ $\{(j, a)$ executes normally$\}$
  Return clean response $o^\star$
end for

2 THE BENCH2ROBUST FRAMEWORK

BENCH2ROBUST is a benchmark-agnostic injection framework that interposes between agent and tool execution, using a domain registry of task-relevant fallback relations and viable paths. Benchmarks following the `tool_call(name, args)` $\rightarrow$ `response` pattern can be integrated via a lightweight adapter; we support τ^2-bench (including Retail-3I/Airline-3I) and BFCL. We model 9 tool-response failures: 6 explicit-signal failures that suppress execution (timeout, rate limit, server error, auth error, malformed response, schema drift) and 3 silent corruptions that alter only the returned observation (partial data, stale values, factual errors). For calls not subject to scenario blocking, each tool call samples $\nu_t \sim \mathrm{Cat}(\boldsymbol{\theta}_j)$ with $P(\text{clean}) = 0.60$; the POMDP formalization appears in Appendix A.1, with the full noise configuration in Appendix A.3.

2.1 SCENARIO-CONTROLLED SOLVABILITY

The central design choice is to assign each training episode an explicit recoverability scenario. In **S1** (original path viable), no task-relevant path is permanently blocked, so the original path remains viable despite stochastic tool failures. In **S2** (switch needed), the task has multiple fallback-equivalent solving paths. We randomly select one path to block; task completion therefore requires switching to a remaining fallback-equivalent path. In **S3** (impossible), all task-relevant viable paths are blocked for the entire episode, so continued tool use cannot complete the task and escalation is the appropriate prescribed behavior. Scenario blocking is deterministic and takes precedence over stochastic failure injection: calls to blocked tools always return a blocked response, whereas calls to unblocked tools remain subject to the per-call stochastic noise distribution with $P(\text{clean}) = 0.60$. Blocking therefore operates outside the stochastic injection budget ($B \leq 5$, $K_{\max} = 2$ consecutive stochastic failures on the same tool), separating episode-level path unavailability from an unlucky sequence of transient failures. Algorithm 1 formalizes this procedure. In the current evaluation, S1 and S2 can be scored through task completion, whereas S3 is used during training because the underlying benchmark evaluators do not assign positive completion credit to unsolvable tasks.

2.2 Alternative Tool Paths

For Retail we add 5 alternative tools (21 total), each providing a different query pattern to the same data (e.g., `search_product_by_name` as alternative to `get_product_details`). For BFCL evaluation, we identify natural equivalence classes within each domain and apply the same S1/S2/S3 blocking. Full specifications in Appendix A.4 (Table 8).

3 Strategy-Aware Training

3.1 Bayesian Tool Memory (BTM)

Random rollouts from a base model rarely contain enough successful recovery behavior to provide useful RL signal, so we supply *Bayesian Tool Memory* (BTM) as structured recovery context during training and optionally at inference. BTM combines (1) Beta-posterior recoverability statistics estimated from training-split rollouts, (2) domain-specific fallback maps with granularity annotations, and (3) recovery constraints for retrying, verification, and escalation.

For each (tool, error type) pair we estimate a Beta-posterior episode-level recovery probability from base-model rollouts on the Retail training split (held-out never used), together with analogous switch-transition beliefs for (source, target) tool pairs; full posterior definitions, sample counts, and the prompt template are in Appendix B.1. These quantities are empirical episode-level recoverability estimates, not causal probabilities that a particular retry or switch action will succeed. During Retail RL training, BTM serves as exploration context that makes retry, switching, verification, and escalation behaviors more likely to appear in rollouts, creating trajectories the optimizer can assign credit to; at inference, the same context supplies domain-specific recovery information.

At inference, dynamic beliefs can either be held *fixed* at their training-split estimates or updated *online* across evaluation episodes. In the online setting, after each completed episode, the Beta counts for the relevant tool–error observations are updated according to whether the trajectory ultimately recovered successfully; the resulting posterior is then used as context for subsequent episodes. Each evaluation run starts from the same training-split posterior initialization, with online counts reset between runs. Episodes are processed in fixed order, and updates use the benchmark outcome observed after each completed trajectory. Policy parameters are never updated during evaluation. Static BTM components are fixed in both settings.

3.2 Reward Design

$$R(\tau) = \begin{cases} 1.0 \cdot \text{eff}(\tau) \cdot \text{rep}(\tau) & \text{if task completed} \\ 0.3 \cdot \frac{\text{matched actions}}{\text{total required}} \cdot \text{rep}(\tau) & \text{otherwise} \end{cases} \tag{1}$$

where $\text{eff}(\tau) = \max(0.3,\ 1.0 - 0.02 \cdot \max(0, |\tau| - 12))$ penalizes long episodes and $\text{rep}(\tau) \in \{0, 0.5, 1.0\}$ penalizes repetition ($\geq$4 identical calls $\to$ 0). The partial-credit term provides dense signal: without it, 38% of training tasks have zero reward variance. Importantly, the reward contains no positive term for correct abstention: S3 episodes cannot pass the benchmark task evaluator. Premature escalation on solvable S1/S2 episodes forfeits completion reward, while the positive prescription to stop after exhausting paths comes from the recovery constraints and exposure to S3 during the curriculum. Accordingly, we treat S3 as a controlled training condition rather than evidence that abstention is independently reward-learned.

3.3 Curriculum and DAPO Training

Five phases progressively introduce harder scenarios, from S1-heavy to a mixed S1/S2/S3 curriculum, with phase advancement gated on diagnostic thresholds (e.g., an S3-conditional escalation rate, not a correctness measure) rather than a claim of mastery. The complete phase schedule and advancement diagnostics, including a training-run measurement of escalation rate before and after S3 exposure, are given in Appendix B.2.

We use DAPO (Yu et al., 2025) with a KL constraint to prevent policy collapse—without it, training collapses within 25–30 iterations as repetition rate exceeds 80%—together with asymmetric clip-

ping, dynamic group filtering, and group-relative advantages over a hybrid clean/injected rollout batch; full hyperparameters are in Appendix B.2.

4 EXPERIMENTAL SETUP

Training and evaluation environments. Interventions are trained and evaluated on **Combined Retail**: 1,339 multi-turn customer-service tasks composed of τ^2-bench retail (114 tasks) (Barres et al., 2025) and Retail-3I (1,225 tasks covering general/ambiguous/changing intents) (Wang et al., 2026a), split into 937 training and 402 held-out tasks. The broader robustness-gap evaluation covers τ^2-bench retail/airline/telecom, Retail-3I, Airline-3I, and BFCL multi-turn (Patil et al., 2025). Cross-benchmark intervention evaluations use Airline-3I (584 tasks), τ^2-bench Telecom (114 tasks), and BFCL multi-turn (200 tasks); target-domain BTM, when enabled, is inference-time context and does not update the Retail-trained RL policy.

Evaluation Modes. (1) **Failure-free (clean):** uncertainty injection is disabled; tool calls return the benchmark environment's original responses without artificial delay, failure, or corruption—i.e., timely, structurally valid, correct, and current with respect to the underlying simulator state. (2) **Inject w/o alt:** injection enabled with the benchmark's standard tool set; scenario control uses only fallback-equivalent paths already present in that tool set. (3) **Inject w/ alt:** injection enabled with augmented tool set and scenario-based blocking.

Models. Qwen3-4B-Thinking-2507 (Yang et al., 2025) is our primary training target; we additionally train Qwen3-8B as a replication. Qwen3-8B/32B/235B (Yang et al., 2025), DeepSeek-V3 (DeepSeek-AI et al., 2025), GLM-4.7 (Team et al., 2025), and MiniMax-M2.5 (Chen et al., 2026a) serve as reference models for the robustness-gap evaluation, spanning four model families and scales from 4B to 235B parameters. Qwen3-235B serves as the user simulator throughout.

5 RESULTS

We first characterize the robustness gap, then evaluate the contributions of runtime recovery context and RL, analyze heterogeneity across failure types, and decompose the intervention.

5.1 THE ROBUSTNESS GAP IS BROAD

We evaluate 7 models from 4 families on 10 task/domain slices under injection (Table 1). Of 70 (model, subset) pairs, 69 degrade (-1.8 to -46.7pp). The gap appears across customer service, airline booking, and function-calling domains. Larger models sometimes lose fewer absolute points, but scaling does not eliminate brittleness: Qwen3-235B still loses 15.8pp on τ^2-retail, while models with 89–91% clean Telecom performance (GLM-4.7, MiniMax-M2.5) lose 20–21pp. BFCL also shows large drops (-11.5 to -39pp). Because task structure and model variants differ across rows, we interpret these results as evidence of broad failure sensitivity rather than as a controlled scaling law; the smallest subsets (n=50–114) carry a $\sim$5–7pp noise floor, so the sole non-degrading cell and several smaller deltas (-1.8 to -5.3pp) are not reliably distinguishable from zero.

5.2 WITHIN-BENCHMARK EFFECTIVENESS

We next evaluate how much robustness comes from structured runtime recovery context and how much additional benefit is associated with RL training. Table 2 reports held-out results (402 tasks not seen during RL training or belief computation) for both the primary 4B target and the Qwen3-8B replication, as point estimates; full mean$\pm$std over 5 evaluation seeds for the 4B rows appears in Appendix Table 12. RL models are trained with BTM as an exploration prior; we evaluate both with and without BTM at inference time to measure internalization.

Structured recovery context provides the largest zero-training gain. Online-updated BTM improves robustness without retraining: 20.1%$\rightarrow$36.9% (+16.8pp w/o alt) and 31.9%$\rightarrow$43.6% (+11.7pp w/ alt). Section 5.4 analyzes which BTM components drive this gain.

Model	τ^2-bench			Retail-3I			Airline-3I			BFCL
	retail (114)	airline (50)	telecom (114)	general (473)	ambig. (473)	changing (279)	general (254)	ambig. (254)	changing (76)	multi-turn (200)
Clean pass rate (%)										
Qwen3-4B-Thinking-2507	56.1	62.0	21.1	71.9	62.0	63.4	65.7	61.0	39.5	31.5
Qwen3-8B	41.2	32.0	25.4	67.0	55.8	54.5	50.0	44.5	21.1	26.0
Qwen3-32B	43.0	30.0	21.1	70.2	58.1	53.8	50.0	51.6	22.4	39.5
Qwen3-235B	57.9	46.0	36.8	72.7	61.1	58.4	67.7	65.8	39.5	59.5
DeepSeek-V3	57.0	50.0	38.6	74.0	62.4	62.0	68.1	68.5	44.7	50.0
GLM-4.7	60.5	64.0	89.5	72.7	60.0	61.7	63.0	66.9	31.6	77.5
MiniMax-M2.5	64.9	68.0	91.2	71.0	61.1	61.7	67.3	68.9	42.1	81.5
Inject pass rate (%)										
Qwen3-4B-Thinking-2507	15.8	52.0	19.3	25.2	19.7	28.7	37.0	31.1	7.9	13.5
Qwen3-8B	17.5	38.0	21.1	41.9	29.8	31.2	46.1	37.8	11.8	14.5
Qwen3-32B	21.1	22.0	16.7	49.1	37.8	41.6	39.4	41.3	11.8	19.0
Qwen3-235B	42.1	40.0	25.4	58.8	44.6	43.0	55.1	55.1	29.0	26.0
DeepSeek-V3	37.7	36.0	29.8	54.1	43.6	43.7	54.7	57.9	39.5	29.0
GLM-4.7	29.8	52.0	68.4	43.8	39.8	39.1	43.3	45.7	19.7	38.5
MiniMax-M2.5	56.1	52.0	71.1	61.3	46.7	52.3	54.3	61.0	34.2	54.5
Δ *(pp)*										
Qwen3-4B-Thinking-2507	−40.4	−10.0	−1.8	−46.7	−42.3	−34.8	−28.7	−29.9	−31.6	−18.0
Qwen3-8B	−23.7	+6.0†	−4.4	−25.2	−26.0	−23.3	−3.9	−6.7	−9.2	−11.5
Qwen3-32B	−21.9	−8.0	−4.4	−21.1	−20.3	−12.2	−10.6	−10.2	−10.5	−20.5
Qwen3-235B	−15.8	−6.0	−11.4	−14.0	−16.5	−15.4	−12.6	−10.6	−10.5	−33.5
DeepSeek-V3	−19.3	−14.0	−8.8	−19.9	−18.8	−18.3	−13.4	−10.6	−5.3	−21.0
GLM-4.7	−30.7	−12.0	−21.1	−29.0	−20.3	−22.6	−19.7	−21.3	−11.8	−39.0
MiniMax-M2.5	−8.8	−16.0	−20.2	−9.7	−14.4	−9.3	−13.0	−7.9	−7.9	−27.0

Table 1: Robustness gap across four task suites (% pass rate, temp=0, inject w/o alt). The first three suites use τ^2-bench-style customer-service infrastructure; BFCL is an independent multi-turn function-calling benchmark. †Small positive delta on τ^2-bench Airline (n=50) is within noise.

Model	Train BTM	Infer BTM	Clean	Inj w/o Alt	Inj w/ Alt
4B	✗	✗	64.3	20.1	31.9
4B	✗	✓	65.3	36.9	43.6
4B + RL	✓	✗	64.5	26.4	38.8
4B + RL	✓	✓	63.9	**40.8**	**45.5**
8B	✗	✗	58.3	33.3	39.4
8B	✗	✓	57.2	39.3	41.2
8B + RL	✓	✗	56.8	37.4	42.2
8B + RL	✓	✓	**59.4**	**41.4**	**44.6**

Table 2: Within-benchmark effectiveness of Qwen3-4B-Thinking-2507 and Qwen3-8B on 402 held-out Combined Retail tasks. Values are point estimates; 5-seed mean±std for the 4B rows is reported in Appendix Table 12. Train BTM denotes structured BTM context used during RL; inference-time BTM uses across-episode online belief updating. Section 5.4 separately compares fixed and online variants on the full 1,339-task set.

RL retains recovery gains without inference-time BTM. To isolate what the policy retains beyond inference-time recovery context, we remove BTM at inference. The RL policy reaches 26.4%/38.8%, versus 20.1%/31.9% for the base model (+6.3/+6.9pp). Base+BTM remains stronger (36.9%/43.6%), while combining both gives the highest performance (40.8%/45.5%), indicating complementary roles for learned recovery behavior and structured runtime context. Qwen3-8B shows the same qualitative pattern.

Full system preserves failure-free performance. RL+BTM clean performance (63.9±1.3%) is within 0.4pp of the base model (64.3±0.9%; Appendix Table 12), showing no measurable clean-task degradation at the resolution of this evaluation.

Gains persist across injection severities. Because our default $P(\text{clean}) = 0.60$ setting is an intentionally severe stress test, we test whether the advantage persists under milder failure rates and different injection budgets (Table 3). As $P(\text{clean})$ increases from 0.60 to 0.95, both systems improve, but RL+BTM retains a +9.0–22.2pp advantage. The gain also remains positive across stochastic injection limits (+19.9–28.3pp).

Setting	Base	RL+BTM	Δ
Vary $P(clean)$; $B = 5$, $K_{\max} = 2$			
$P(\text{clean}) = 0.60$	21.6	43.8	+22.2
$P(\text{clean}) = 0.80$	35.3	56.0	+20.7
$P(\text{clean}) = 0.90$	49.0	61.4	+12.4
$P(\text{clean}) = 0.95$	53.2	62.2	+9.0
Vary injection limits; $P(clean) = 0.60$			
$B = 3$, $K_{\max} = 1$	23.9	52.2	+28.3
$B = 5$, $K_{\max} = 2$	21.6	43.8	+22.2
$B = 7$, $K_{\max} = 3$	21.9	41.8	+19.9

Table 3: Sensitivity to injection severity on 402 held-out Combined Retail tasks (seed 42, inject w/o alt). B is the maximum number of stochastic injections per episode; $K_{\max}$ is the maximum number of consecutive failures on the same tool. Values are pass rates (%).

	Pass Rate (%)			Incremental Δ (pp)	
Error Type	Base	+BTM	RL+BTM	ΔBTM	ΔRL
Service/response explicit errors					
timeout	12.4	32.5	38.1	**+20.0**	+5.6
rate limit	15.2	33.1	40.5	**+18.0**	+7.3
server error	6.2	31.2	34.8	**+25.0**	+3.6
malformed response	11.5	36.8	37.8	**+25.3**	+1.0
Auth/schema explicit errors					
auth error	13.5	25.9	36.1	**+12.4**	+10.2
schema drift	14.9	27.5	40.0	**+12.6**	+12.5
Silent corruptions					
partial	20.6	29.1	33.6	**+8.5**	+4.5
stale	28.9	31.6	37.0	+2.7	**+5.4**
factual error	33.8	29.2	39.8	−4.6	**+10.5**

Table 4: Pass rate by failure type under injection without alternatives. BTM denotes the inference-time context intervention; RL+BTM is the full system. Incremental deltas are $\Delta\text{BTM} = (+\text{BTM}) - \text{Base}$ and ΔRL denotes the additional difference associated with RL training when inference-time BTM is held present. Values are point estimates. BTM uses the same online-updated inference protocol as Table 2.

Learned recovery behavior shows small positive point-estimate gains across domains. Without retraining or target-domain BTM, the Retail-trained RL policy improves injected pass rate by +2.7pp on Airline-3I and +1.5pp on BFCL; adding target-domain BTM increases the gains to +9.0pp and +5.0pp, respectively. On Telecom, gains are within noise, and the 4B model is already near its clean-performance floor. This pattern is consistent with RL learning partially reusable recovery behavior while runtime context supplies domain-specific recovery information (Appendix D.2).

5.3 BTM and RL Address Different Failure Regimes

The aggregate gains in Section 5.2 mask substantial heterogeneity across failure types (Table 4). BTM provides its largest gains on timeout, rate-limit, server, and malformed-response failures (+18.0 to +25.3pp), where explicit failure signals make the supplied recovery guidance directly actionable.

The relative contribution of BTM and RL differs for the remaining error types. The RL-associated increment is +10.2/+12.5pp on auth and schema errors and +10.5pp on factual corruption, while BTM's incremental gain is smaller and becomes negative for factual corruption (-4.6pp). We treat this pattern as descriptive heterogeneity across injected error types rather than evidence of distinct underlying failure mechanisms.

5.4 BTM Decomposition and Training Recipe Comparison

(a) Training-recipe comparison

		Inj (%)	
Method	**Clean**	**No Alt**	**Alt**
Base, no FT	65.3	23.2	26.4
vanilla SFT	65.3	24.0	24.5
vanilla GRPO	64.9	23.0	24.4
Ours (RL+BTM)	65.3	**45.9**	**47.9**

(b) BTM structural decomposition

			Pass (%)	
Variant	**S**	**D**	**Fix.**	**Onl.**
Base (no belief)	✗	✗	23.2	–
Dynamic only	✗	✓	24.9	–
Static + uniform init.	✓	✗	33.7	40.8
Full	✓	✓	37.8	**41.6**

Table 5: (a) Training-recipe comparison on the full 1,339-task Combined Retail set. Only *Ours* uses BTM (RL exploration prior + inference-time context); vanilla SFT/GRPO use no BTM at inference. (b) BTM structural decomposition on the same 1,339-task set (inject w/o alt). S/D = Static recovery structure / training-derived dynamic priors; Fix./Onl. = dynamic posterior beliefs held fixed at their training-split estimates vs. updated after each completed evaluation episode using observed trajectory recovery outcomes. Both panels report point estimates with no seed-level variance. For the "Static + uniform init." variant, dynamic probabilities are initialized uniformly rather than from training rollouts; in the online condition, these uniform initial values are updated from subsequent evaluation trajectories.

With fixed context, static recovery structure contributes most: Static + uniform init. improves pass rate by +10.5pp, whereas dynamic priors alone add +1.7pp. Online updating raises the former to 40.8%, close to Full BTM at 41.6%, showing that trajectory-based adaptation can substantially strengthen initially uninformative beliefs. Because the static condition combines generic recovery constraints with domain-specific fallback relations, their individual contributions remain unresolved.

Standard fine-tuning alone does not improve robustness: vanilla SFT and GRPO remain within 1pp of the base model under injection without alternatives (Table 5(a)). Our full recipe reaches 45.9%. On the same full-set condition, this is +8.1pp over Base+Full BTM with fixed beliefs and +4.3pp with online beliefs. Because this evaluation includes the 937 training tasks, we treat these differences only as a consistency check; the held-out estimate of the RL-associated gain is the +6.3pp result in Section 5.2.

6 Discussion

Controlled solvability and complementary recovery mechanisms. Scenario control disentangles environment recoverability from policy behavior, while the results show complementary contributions from runtime recovery context and trained recovery behavior.

Implications and boundaries. BTM should be viewed primarily as structured recovery context: static structure is most useful when context is fixed, while online updating increases the value of dynamic beliefs. The evidence is also strongest for retry and switch, given the reward design's incomplete abstention signal (Section 3.2). For deployment, the main implication is practical rather than architectural: tool-failure robustness should be evaluated separately from clean task success, and jointly with efficiency, because recovery improves pass rate while increasing token use. Recovery context increases inference cost relative to the base model, but RL does not uniformly add further overhead: with alternatives, RL+BTM uses fewer tokens and steps than Base+BTM while achieving a higher pass rate (Appendix D.3).

Limitations. Our failures are simulated rather than incident-derived, although gains persist under milder injection settings. S3 receives no positive reward for correct stopping, so we do not establish independently learned abstention. Finally, each RL model is represented by a single training run; evaluation seeds capture rollout and injection randomness, not training-seed variance.

7 Related Work

Tool-Using LLM Agents and Agentic RL. Tool-augmented LLMs have been studied via self-supervised API training (Schick et al., 2023), retrieval-augmented tool use (Patil et al., 2024), large-scale API benchmarks (Qin et al., 2024; Li et al., 2023), and multi-turn task completion (Yao et al., 2025), with evaluation largely centered on nominal tool execution rather than controlled tool-failure recovery. On the optimization side, GRPO (Shao et al., 2024) drops the critic via group-relative advantages, and DAPO (Yu et al., 2025) adds asymmetric clipping and dynamic filtering; we build on DAPO with KL stabilization and a scenario-controlled curriculum for multi-turn recovery.

Robustness Benchmarks for Tool-Using Agents. Recent work studies tool blocking (Liu et al., 2026), noisy environments (Wang et al., 2026b; Gupta, 2026), dynamic replanning (Zhu et al., 2026), and imperfect APIs (Kim et al., 2026). ToolBench-X (Tian et al., 2026) is closest to our injection design; Bench2Robust instead organizes failures by observability and controls whether recovery requires retrying, switching, or facing no viable path, enabling both controlled evaluation and training (Appendix C).

Training for Tool-Use Robustness. NoisyAgent (Chen et al., 2026b), the closest concurrent work, injects RL-rollout noise with curriculum scheduling. RobustBench-TC (Zhou et al., 2026) trains via domain-randomized GRPO but targets observation-, action-, and reward-level perturbations, excluding tool-response failures from training and probing them only zero-shot with a single, 100%-rate injection at the first call. Neither controls, during training or evaluation, whether recovery requires staying on the original path, switching, or facing no viable path; Bench2Robust adds this recoverability structure and explicit alternatives, letting the regimes be analyzed separately (our abstention signal remains reward-incomplete, Section 3.2). PALADIN (Vuddanti et al., 2026) trains on failure-injected, recovery-annotated trajectories with exemplar retrieval but does not control scenario solvability; ToolCritic (Hamad et al., 2025) corrects errors via a separate critic rather than training the policy end-to-end. We instead train a single policy across episodes that each demand a different recovery regime.

Recovery, Abstention, and Posterior-Based Memory. At inference time, ReAct (Yao et al., 2023) and Reflexion (Shinn et al., 2023) add retry/reflection, though LLMs cannot reliably self-correct without external feedback (Huang et al., 2024). Abstention has been studied at the knowledge level (R-Tuning (Zhang et al., 2024)) and the user-intent level (Learning to Ask (Wang et al., 2025)); we formulate an analogous stopping condition at the *tool-response* level, subject to the same reward-design limitation (Section 3.2). Bayesian-Agent (Wu et al., 2026) maintains a posterior over reusable prompt-side skills between episodes; BTM instead supplies per-tool$\times$error posteriors as runtime context, with optional across-episode online updating, complementary rather than competing uses of a similar idea. Risk-oriented testing shows agents may fail to act on their own risk knowledge (Ruan et al., 2024; Tang et al., 2025); we study whether structured context and policy training together improve on this.

8 Conclusion

We presented Bench2Robust, a framework for studying tool-failure robustness as a recovery-policy problem rather than persistence alone. Scenario-controlled solvability separates original-path-viable, switch-required, and no-viable-path regimes, exposing a broad robustness gap across seven models. On held-out Retail tasks using the benchmark's original tool set, structured recovery context improves robustness by +16.8pp, RL retains a +6.3pp gain without inference-time context, and combining the two yields a +20.7pp improvement over the base model without measurable clean-task degradation. Overall, robust tool use appears to benefit from combining structured runtime recovery context with recovery behavior acquired through training.

AI USE STATEMENT

Generative AI tools were used to assist with editing and polishing the manuscript for clarity, concision, and readability. The authors reviewed and verified all AI-assisted edits and take full responsibility for the final content of the paper.

REPRODUCIBILITY STATEMENT

All experiments use publicly available benchmarks (τ^2-bench, Retail-3I, Airline-3I, BFCL) and open-weight base models (Qwen3-4B-Thinking-2507, Qwen3-8B). Appendix B.2 gives the complete training configuration; Appendix A.3 gives the per-tool noise distribution, injection budget, and consecutive-failure limits; Appendix A.2 defines the nine noise types; Appendix A.4 enumerates the alternative tools; and Appendix B.1 gives the verbatim BTM prompt template. We will release the injection framework, domain registry, belief-computation scripts, and evaluation harness.

ETHICS STATEMENT

This work studies failure recovery in tool-using agents using only simulated tool failures in sandboxed benchmarks; no real APIs or user data are involved. We see a primarily beneficial safety application: agents that recover from tool failures without fabricating completion once recovery paths are exhausted could reduce unwarranted actions in deployment. Because our reward does not positively reinforce correct stopping (Section 3.2), this remains a design goal rather than a demonstrated property of the trained policy. Two risks merit note: training for persistence could encourage excessive retrying against real services, though our efficiency discount and repetition penalty are intended to bound this; and an escalation threshold appropriate in one domain may transfer poorly to another, since the stopping prescription is supplied by the prompt rather than learned from an abstention reward. Our Telecom results illustrate this transfer limitation: recovery training does not remedy a capability ceiling, and such systems should not be presumed robust outside their evaluated domains.

REFERENCES

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=Yacmpz84TH.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, dahai li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=dHng2O0Jjr.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive APIs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=tBRNC6YemY.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik R Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=roNSXZpUDN.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-bank: A comprehensive benchmark for tool-augmented LLMs. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://openreview.net/forum?id=o2HBfgY20b.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982.

Ziyi Wang, Yuxuan Lu, Yimeng Zhang, Pei Chen, Ziwei Dong, Jing Huang, Jiri Gesi, Xianfeng Tang, Chen Luo, Qun Liu, Yisi Sang, Hanqing Lu, Manling Li, Jin Lai, and Dakuo Wang. Trajectory2Task: Training robust tool-calling agents with synthesized yet verifiable data for complex user intents. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens, editors, *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 44021–44044, San Diego, California, United States, July 2026a. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.2037. URL https://aclanthology.org/2026.acl-long.2037/.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=2GmDdhBdDk.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, YuYue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=2a36EMSSTp.

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.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan

Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. Deepseek-v3 technical report, 2025. URL https://arxiv.org/abs/2412.19437.

GLM Team, Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, Kedong Wang, Lucen Zhong, Mingdao Liu, Rui Lu, Shulin Cao, Xiaohan Zhang, Xuancheng Huang, Yao Wei, Yean Cheng, Yifan An, Yilin Niu, Yuanhao Wen, Yushi Bai, Zhengxiao Du, Zihan Wang, Zilin Zhu, Bohan Zhang, Bosi Wen, Bowen Wu, Bowen Xu, Can Huang, Casey Zhao, Changpeng Cai, Chao Yu, Chen Li, Chendi Ge, Chenghua Huang, Chenhui Zhang, Chenxi Xu, Chenzheng Zhu, Chuang Li, Congfeng Yin, Daoyan Lin, Dayong Yang, Dazhi Jiang, Ding Ai, Erle Zhu, Fei Wang, Gengzheng Pan, Guo Wang, Hailong Sun, Haitao Li, Haiyang Li, Haiyi Hu, Hanyu Zhang, Hao Peng, Hao Tai, Haoke Zhang, Haoran Wang, Haoyu Yang, He Liu, He Zhao, Hongwei Liu, Hongxi Yan, Huan Liu, Huilong Chen, Ji Li, Jiajing Zhao, Jiamin Ren, Jian Jiao, Jiani Zhao, Jianyang Yan, Jiaqi Wang, Jiayi Gui, Jiayue Zhao, Jie Liu, Jijie Li, Jing Li, Jing Lu, Jingsen Wang, Jingwei Yuan, Jingxuan Li, Jingzhao Du, Jinhua Du, Jinxin Liu, Junkai Zhi, Junli Gao, Ke Wang, Lekang Yang, Liang Xu, Lin Fan, Lindong Wu, Lintao Ding, Lu Wang, Man Zhang, Minghao Li, Minghuan Xu, Mingming Zhao, Mingshu Zhai, Pengfan Du, Qian Dong, Shangde Lei, Shangqing Tu, Shangtong Yang, Shaoyou Lu, Shijie Li, Shuang Li, Shuang-Li, Shuxun Yang, Sibo Yi, Tianshu Yu, Wei Tian, Weihan Wang, Wenbo Yu, Weng Lam Tam, Wenjie Liang, Wentao Liu, Xiao Wang, Xiaohan Jia, Xiaotao Gu, Xiaoying Ling, Xin Wang, Xing Fan, Xingru Pan, Xinyuan Zhang, Xinze Zhang, Xiuqing Fu, Xunkai Zhang, Yabo Xu, Yandong Wu, Yida Lu, Yidong Wang, Yilin Zhou, Yiming Pan, Ying Zhang, Yingli Wang, Yingru Li, Yinpei Su, Yipeng Geng, Yitong Zhu, Yongkun Yang, Yuhang Li, Yuhao Wu, Yujiang Li, Yunan Liu, Yunqing Wang, Yuntao Li, Yuxuan Zhang, Zezhen Liu, Zhen Yang, Zhengda Zhou, Zhongpei Qiao, Zhuoer Feng, Zhuorui Liu, Zichen Zhang, Zihan Wang, Zijun Yao, Zikang Wang, Ziqiang Liu, Ziwei Chai, Zixuan Li, Zuodong Zhao, Wenguang Chen, Jidong Zhai, Bin Xu, Minlie Huang, Hongning Wang, Juanzi Li, Yuxiao Dong, and Jie Tang. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models, 2025. URL https://arxiv.org/abs/2508.06471.

MiniMax: Aili Chen, Aonian Li, Baichuan Zhou, Bangwei Gong, Binyang Jiang, Boji Dan, Changqing Yu, Chao Wang, Cheng Ma, Cheng Zhong, Cheng Zhu, Chengjun Xiao, Chengyi Yang, Chengyu Du, Chenyang Zhang, Chi Zhang, Chuangyi Huang, Chunhao Zhang, Chunhui Du, Chunyu Zhao, Congchao Guo, Da Chen, Deming Ding, Dianjun Sun, Dongyu Zhang, Enhui Yang, Fei Yu, Guang Zheng, Guodong Zheng, Guohong Li, Haichao Zhu, Haigang Zhou, Haimo Zhang, Han Ding, Hao Zhang, Haohai Sun, Haolin Lyu, Haonan Lu, Haoyu Wang, Huajie Shi, Huiyang Li, Jiacheng Chen, Jian Zhang, Jiaqi Zhuang, Jiaren Cai, Jiaxin Pan, Jiayao Li, Jiayuan Song, Jichuan Zhang, Jie Wang, Jihao Gu, Jin Zhu, Jingwei Dong, Jingyang Li, Jingyu Zhang, Jingze Zhuang, Jinhao Tian, Jinli Liu, Jinyi Hu, Jun Tao, Jun Zhang, Junbin Ruan, Junhao Xu, Junjie Yan, Junteng Liu, Junxian He, Kang Xu, Ke Ji, Ke Yang, Kecheng Xiao, Keyu Duan, Keyu Li, Le Han, Letian Ruan, Li Yuan, Lianfei Yu, Liheng Feng, Lijie Mo, Lin Li, Lingye Bao, Lingyu Yang, Lingyuan Zhou, Loki, Lu Chen, Lunbin Ceng, Ming Li, Ming Zhong, Mingliang Tao, Mingyuan Chi, Mujie Lin, Nan Hu, Ningxin Chen, Peiyin Zhu, Peng Gao, Pengcheng Gao, Pengfei Li, Penglin Li, Pengyu Zhao, Qibin Ren, Qidi Xu, Qihan Ren, Qile Li, Qin Wang, Quanliang Chen, Qunhong Ceng, Rong Tian, Rui Dong, Ruitao Leng, Ruize Zhang, Shanqi Liu, Shaoyu Chen, Sheng Jia, Shun Yao, Shuoran Zhao, Shuqi Yu, Sichen Li, Sicheng Pan, Songquan Zhu, Tengfei Li, Tian Xie, Tiancheng Qin, Tianrun Liang, Wei Liu, Weiqi Xu, Weitao Li, Weixiang Chen, Weiyu Cheng, Weiyu Zhang, Wenhu Chen, Wenqian Zhao, Xiancai Chen, Xiangjun Song, Xiangyuan Wang, Xiao Luo, Xiao Su, Xiaobo Li, Xiaodong Han, Xiaojie Wu, Xihao Song, Xingyi Han, Xinyu Guan, Xuan Lu, Xun Zou, Xunhao Lai, Xutong Li, Yan Gong, Yang Wang, Yang Xu, Yangsen Wang, Ye Tang, Yicheng Chen, Yinran Qiu, Yiqi Shi, Yiting Guo, Yiwen Huang, Yixuan Wang, Yongyi Hu, Yu Gao, Yu Zhang, Yuanxiang Ying, Yuanzhen Zhang, Yubo Wang, Yuchen Song, Yufeng Yang, Yuhang Meng, Yuhang Miao, Yuhao Li, Yujie Liu, Yulin Hu, Yunan Huang, Yunji Li, Yunyi Huang, Yusen Zhang, Yusu Hong, Yutao Xie, Yutong Zhang, Yuwen Liao, Yuxuan Shi, Yuze Wenren, Zebin Li, Zehan Li, Zejian Luo, Zeyu Jin, Zeyuan Sun, Zhanpeng Zhou, Zhaochen Su, Zhendong Li, Zhengmao Zhu, Zhengyuan Peng, Zhenhua Fan, Zhi Zhang, Zhichao Xu, Zhiheng Lv, Zhikang Xu, Zhitao He, Zhiwei He, Zhongyuan Li, Zibo

Gao, Zijia Wu, Zijian Song, Zijian Zhou, Zijun Sun, Zishan Huang, Ziying Chen, and Ziyue Ge. The minimax-m2 series: Mini activations unleashing max real-world intelligence, 2026a. URL https://arxiv.org/abs/2605.26494.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Jiayu Liu, Qihan Lin, Cheng Qian, Rui Wang, Emre Can Acikgoz, Xiaocheng Yang, Jiateng Liu, Zhenhailong Wang, Xiusi Chen, Heng Ji, and Dilek Hakkani-Tür. Planbench-xl: Evaluating long-horizon planning of llm tool-use agents in large-scale tool ecosystems, 2026. URL https://arxiv.org/abs/2606.22388.

Ruipeng Wang, Yuxin Chen, Yukai Wang, Chang Wu, Junfeng Fang, Xiaodong Cai, Qi GU, Hui Su, An Zhang, Xiang Wang, Xunliang Cai, and Tat-Seng Chua. Agentnoisebench: Benchmarking robustness of tool-using LLM agents under noisy condition. In *Forty-third International Conference on Machine Learning*, 2026b. URL https://openreview.net/forum?id=ORVJCJczoG.

Aayush Gupta. Reliabilitybench: Evaluating llm agent reliability under production-like stress conditions, 2026. URL https://arxiv.org/abs/2601.06112.

Dongsheng Zhu, Xuchen Ma, Yucheng Shen, Xiang Li, Yukun Zhao, Shuaiqiang Wang, Lingyong Yan, and Dawei Yin. When tools fail: Benchmarking dynamic replanning and anomaly recovery in llm agents, 2026. URL https://arxiv.org/abs/2606.05806.

Doyoung Kim, Zhiwei Ren, Jie Hao, Zhongkai Sun, Lichao Wang, Xiyao Ma, Zack Ye, Xu Han, Jun Yin, Heng Ji, Wei Shen, Xing Fan, Benjamin Yao, and Chenlei Guo. Beyond perfect apis: A comprehensive evaluation of llm agents under real-world api complexity, 2026. URL https://arxiv.org/abs/2601.00268.

Yang Tian, Zhengpeng Shi, Yu Zhou, and Bo Zhao. Beyond function calling: Benchmarking tool-using agents under tool-environment unreliability, 2026. URL https://arxiv.org/abs/2606.25819.

Yuxin Chen, Xiaodong Cai, Junfeng Fang, Zhuowen Han, Yu Wang, Yaorui Shi, Yi Zhang, Qi Gu, Xunliang Cai, Xiang Wang, An Zhang, and Tat-Seng Chua. Learning to act under noise: Enhancing agent robustness via noisy environments, 2026b. URL https://arxiv.org/abs/2605.27209.

Xiaolin Zhou, Aojie Yuan, Zheng Luo, Zipeng Ling, Xixiao Pan, Yicheng Gao, Haiyue Zhang, Jiate Li, Shuli Jiang, Prince Zizhuang Wang, Zixuan Zhu, Jinbo Liu, Ryan A. Rossi, Hua Wei, and Xiyang Hu. When simulation lies: A sim-to-real benchmark and domain-randomized RL recipe for tool-use agents. In *Workshop on Failure Modes of Agentic AI at ICML 2026*, 2026. URL https://openreview.net/forum?id=9hC5wTcZ4u.

Sri Vatsa Vuddanti, Aarav Shah, Satwik Kumar Chittiprolu, Tony Song, Sunishchal Dev, Kevin Zhu, and Maheep Chaudhary. PALADIN: Self-correcting language model agents to cure tool-failure cases. In *LLM-based Multi-Agent Systems: Towards Responsible, Reliable, and Scalable Agentic Systems*, 2026. URL https://openreview.net/forum?id=NVTtoO297p.

Hassan Hamad, Yingru Xu, Liang Zhao, Wenbo Yan, and Narendra Gyanchandani. Toolcritic: Detecting and correcting tool-use errors in dialogue systems, 2025. URL https://arxiv.org/abs/2510.17052.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=vAElhFcKW6.

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://openreview.net/forum?id=IkmD3fKBPQ.

Hanning Zhang, Shizhe Diao, Yong Lin, Yi Fung, Qing Lian, Xingyao Wang, Yangyi Chen, Heng Ji, and Tong Zhang. R-tuning: Instructing large language models to say 'I don't know'. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7113–7139, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.394. URL https://aclanthology.org/2024.naacl-long.394/.

Wenxuan Wang, Shi Juluan, Zixuan Ling, Yuk-Kit Chan, Chaozheng Wang, Cheryl Lee, Youliang Yuan, Jen-tse Huang, Wenxiang Jiao, and Michael R. Lyu. Learning to ask: When LLM agents meet unclear instruction. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng, editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 21773–21784, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1104. URL https://aclanthology.org/2025.emnlp-main.1104/.

Xiaojun Wu, Cehao Yang, Honghao Liu, Xueyuan Lin, Wenjie Zhang, Zhichao Shi, Xuhui Jiang, Chengjin Xu, Jia Li, and Jian Guo. Bayesian-agent: Posterior-guided skill evolution for llm agent harnesses, 2026. URL https://arxiv.org/abs/2606.08348.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. Identifying the risks of LM agents with an LM-emulated sandbox. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=GEcwtMk1uA.

Yuzhi Tang, Tianxiao Li, Elizabeth Li, Chris J. Maddison, Honghua Dong, and Yangjun Ruan. LM agents may fail to act on their own risk knowledge. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=OeYdS51k8F.

A Failure Injection Details

A.1 Bench2Robust Injection POMDP Formalization

We formalize the injected environment as a POMDP $\mathcal{M}_{\text{inj}} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, T_{\text{inj}}, Z_{\text{inj}}, R, \gamma)$ that shares the clean benchmark's reward R, but whose transition and observation kernels depend on which noise type ν is realized for a call to tool j.

We distinguish noise types by whether the underlying tool call is committed. Let $\mathcal{N}_{\text{exec}}$ denote the six *explicit-signal* failures (timeout, rate limit, server error, auth error, malformed response, schema drift) and $\mathcal{N}_{\text{silent}}$ the three *silent* corruptions (partial data, stale values, factual errors). For $\nu \in \mathcal{N}_{\text{exec}}$, and likewise for episode-persistent scenario blocking (Section 2.1), the call does not execute: the agent receives only a failure message, and the underlying state is unaffected. For $\nu = \text{clean}$ or $\nu \in \mathcal{N}_{\text{silent}}$, the call executes normally and only the reported observation may be altered:

$$T_{\text{inj}}(s_{t+1} \mid s_t, a_t, \nu) = \begin{cases} T_{\text{clean}}(s_{t+1} \mid s_t, a_t) & \nu \in \{\text{clean}\} \cup \mathcal{N}_{\text{silent}} \\ \delta(s_{t+1} = s_t) & \nu \in \mathcal{N}_{\text{exec}} \text{ or scenario-blocked} \end{cases} \tag{2}$$

$$Z_{\text{inj}}(o_t \mid s_t, a_t, \nu) = \begin{cases} C_\nu(Z_{\text{clean}}(s_t, a_t)) & \nu \in \{\text{clean}\} \cup \mathcal{N}_{\text{silent}} \\ C_\nu(\cdot) & \nu \in \mathcal{N}_{\text{exec}} \text{ or scenario-blocked} \end{cases} \tag{3}$$

where C_{clean} is the identity map, δ denotes a no-op transition, and for $\nu \in \mathcal{N}_{\text{exec}}$ or scenario blocking, $C_\nu(\cdot)$ is the corresponding failure or blocked-response message rather than a corruption of an executed response.

A practical consequence is that retrying after an explicit-signal failure or a scenario-blocked call (e.g., a `cancel_order` call that returns a timeout) does not risk a duplicate state mutation, whereas retrying after a silent corruption re-executes the tool and can return an independently corrupted value. This is a simplifying assumption rather than a model of transactional failure semantics (e.g., a request that fails only after partially committing side effects); we do not simulate that intermediate case.

For calls not subject to scenario blocking, each tool call to tool j independently samples $\nu_t \sim \text{Cat}(\boldsymbol{\theta}_j)$ with $P(\text{clean}) = 0.60$; the remaining 0.40 is spread over the nine failure modes in $\mathcal{N}_{\text{exec}} \cup \mathcal{N}_{\text{silent}}$. The complete scenario-controlled injection procedure, Algorithm 1, is given in Section 2.1.

A.2 Noise Types

Table 6 details the nine noise types used by Bench2Robust, organized by observability.

Noise Type	Injected Response
Observable Errors — explicit error signal visible to the agent	
Timeout	"Error: request timed out after 30s"
Rate Limit	HTTP 429 Too Many Requests
Server Error	HTTP 500/502/503 Internal Server Error
Malformed Response	JSON parse failure / garbled output
Auth Error	HTTP 401/403 Forbidden
Schema Drift	"Unknown field 'X' / required field missing"
Silent Errors — valid response with corrupted decision-critical fields	
Partial	Critical fields → empty ([], {}, "")
Stale	State fields backdated (e.g., "delivered" → "pending")
Factual Error	Values ±10–25% plausible perturbation

Table 6: Nine noise types grouped by observability. Observable errors produce unambiguous error messages; silent errors return well-formed responses with corrupted semantics that require verification or cross-referencing to detect.

	Observable Errors						Silent Errors				Limits	
	Timeout	Rate Lim.	Server	Malform.	Auth	Schema	Partial	Stale	Factual	Clean	Budget	Consec.
$P(\nu)$	0.06	0.05	0.05	0.04	0.03	0.03	0.05	0.04	0.05	0.60	5	2

Table 7: Per-tool noise configuration ($\boldsymbol{\theta}_j$). Each tool call is clean with probability 0.60; otherwise one of nine failure modes is sampled. Budget and consecutive-failure limits apply only to stochastic injection, not episode-persistent scenario blocking. This severe stress-test setting is not an estimate of real API failure rates.

Alternative Tool	Equivalent To	Different Path
`search_product_by_name`	`get_product_details`	Lookup by name vs. ID
`get_item_by_product_options`	`get_item_details`	Find item by options vs. ID
`get_orders_for_user`	`get_order_details`	Retrieve all orders at once (summary)
`get_user_by_order`	`find_user_id_by_*`	Authenticate via order number
`check_order_item_availability`	`get_product` + matching	One-shot exchange availability check

Table 8: Retail alternative tools (5 added; 21 tools total).

A.3 Injection Configuration

Table 7 reports the per-tool noise distribution $\boldsymbol{\theta}_j$ used throughout training and evaluation, together with the stochastic injection budget and consecutive-failure limits referenced in Section 2.1.

A.4 Alternative Tool Paths

For Retail we add five alternative tools to the standard 16-tool environment (21 tools total when `with_alternatives=True`). Each provides a different query path to equivalent task-relevant information.

B BTM and Training Details

B.1 Bayesian Tool Memory Posterior and Prompt Template

For each (tool j, error type e) pair, recovery probability is modeled as

$$p_{j,e}^{\text{rec}} \sim \text{Beta}(\alpha_{j,e}, \beta_{j,e}),$$

with posterior mean $\hat{P} = \alpha/(\alpha+\beta)$, computed for the Retail intervention from base-model rollouts on the Retail training split (937 tasks, 3–5 seeds; held-out never used). This is an *episode-level recoverability* estimate: "when tool j hits error e, how often does the episode still succeed?"

Switch beliefs $\hat{P}(\text{switch} \mid j \to k)$ are defined analogously for (source, target) pairs. These are observed recovery rates following $j \to k$ transitions, not causal switch-success probabilities, and carry higher uncertainty ($n = 6$–20) due to sparser data and selection bias.

Figure 2 shows the BTM block injected into the system prompt. Posterior recoverability statistics are presented together with predefined fallback structure and recovery constraints; the model chooses actions from this combined context. The template is instantiated per domain: tool names, belief values, and fallback maps are populated from the target environment's configuration.

B.2 Training Configuration and Curriculum

Table 9 details the five-phase scenario curriculum, and Table 10 lists the complete DAPO training configuration referenced in Section 3.3.

In the 4B training run, escalation rises from 23.7% before S3 exposure to 33.3% in later phases, with average turns stabilizing at 10.7. We report this only as a curriculum diagnostic, not as an

```
<tool_reliability>
Tool recovery beliefs (Beta posterior, from historical data):
{TOOL_NAME_1}:
P(episode recovery | error):  timeout={p}, rate_limit={p},
server_error={p}, auth_error={p}, schema_drift={p},
malformed_response={p}, partial={p}, stale={p},
factual_error={p}
Alternative:  {ALT_TOOL} (P={p}, {GRANULARITY})
{TOOL_NAME_2}:
P(episode recovery | error):  timeout={p}, ...
Alternative:  {ALT_TOOL} (P={p}, {GRANULARITY})
...

How to use:  Compare P(recovery) vs P(switch) for your error.
High P(recovery) -> retry this tool.
Low P(recovery) + high P(switch) -> switch to alternative.
After switching, verify returned data has needed fields.
If both low -> ask the user or transfer.

Alternative tool paths (when primary tool fails):
- {PRIMARY} -> {FALLBACK} ({GRANULARITY NOTE})
- ...

Critical constraints:
1.  If a tool fails, retry at least once before other options.
2.  After switching, verify returned data provides ALL required
fields at the right granularity.
3.  Before any irreversible action, confirm every required field
from actual tool responses.  Do not guess missing fields.
4.  For ANY price calculation, use the calculate tool.
5.  After calling a write tool, check the response for errors.
6.  Only transfer after exhausting retry and alternative paths.
</tool_reliability>
```

Figure 2: BTM system prompt template. Placeholders are populated per domain with recovery beliefs, fallback maps, and shared recovery constraints.

Phase	Focus	Max Inj.	S1/S2/S3	Advance when
1	S1: retry	2	100/0/0	Avg. reward > 0.25 for 3 iters
2	Discover alternatives	2	80/20/0	Switch $> 25\%$ & pass $> 25\%$
3	S2: switch	5	60/40/0	Switch $> 25\%$ & pass $> 25\%$
4	S3: no viable path	5	40/50/10	S3 escalation $> 50\%$, turns < 15, no S1/S2 regression
5	Mixed	5	30/45/25	Early stopping

Table 9: Scenario curriculum and phase-advancement diagnostics. Phase 4 uses the S3-conditional transfer rate as a training diagnostic, not an abstention correctness measure; Phase 5 early-stops on sustained over-escalation.

abstention-correctness measure. Without the KL constraint, training collapses within 25–30 iterations as repetition rate exceeds 80%.

C Error-Type Coverage of Prior Robustness Benchmarks

Table 11 situates our noise taxonomy against prior robustness benchmarks, grouped by observability.

Parameter	Value
Base model	Qwen3-4B-Thinking-2507
Algorithm	DAPO (group-relative with asymmetric clip + KL stabilization)
KL loss	low_var_kl, coefficient 0.02
Asymmetric clipping	$\epsilon_{\text{low}} = 0.20$, $\epsilon_{\text{high}} = 0.28$
Dynamic group filtering	Groups with $\sigma < 0.01$ masked
Learning rate	2×10^{-6} (constant)
Global batch size	256 (16 prompts $\times$ 16 rollouts)
Rollouts per prompt	16 (first 8 clean + last 8 noisy, unified normalization)
Max response tokens	8192
Max steps per episode	25
Context length	32768
Epochs	3
Training data	937 Combined Retail tasks
Reward	partial_credit $\times 0.3$ (fail) or $1.0\times$ efficiency (pass)
Repetition penalty	$3\times$ same call $\rightarrow 0.5$; $4\times \rightarrow 0.0$
Efficiency discount	$\max(0.3,\ 1.0 - 0.02\max(0, \text{turns} - 12))$
Curriculum	Phase 1–5 (Phases 1–4 auto-advance; Phase 5 early stopping)
Bayesian Tool Memory	Beta posteriors computed from training-split rollouts
Hardware	8×H100 80GB
User simulator	Qwen3-235B via AWS Bedrock
Framework	slime

Table 10: Full DAPO training configuration.

	Observable Errors						Silent Errors		
Benchmark	Timeout	Rate Lim.	Server Err.	Malformed	Auth Err.	Schema Dr.	Partial	Stale	Factual
ToolBench-X (Tian et al., 2026)	✓	✗	✓	✗	✗	✓	✗	✗	✓
RobustBench-TC (Zhou et al., 2026)	✓	✓	✓	✓	✓	✓	✗	✗	✗
AgentNoiseBench (Wang et al., 2026b)	✓	✗	✓	✓	✗	✗	✓	✗	✓
NoisyAgent (Chen et al., 2026b)	✓	✗	✓	✓	✗	✗	✓	✗	✓
ToolMaze (Zhu et al., 2026)	✓	✓	✗	✗	✗	✗	✓	✗	✓
WildAGTEval (Kim et al., 2026)	✗	✗	✓	✗	✗	✗	✓	✗	✗
ReliabilityBench (Gupta, 2026)	✓	✓	✗	✓	✗	✓	✓	✗	✗
PlanBench-XL (Liu et al., 2026)	✗	✗	✓	✗	✗	✗	✗	✗	✗
PALADIN (Vuddanti et al., 2026)	✓	✗	✓	✗	✗	✗	✗	✗	✓
BENCH2ROBUST (Ours)	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 11: Error-type coverage across observable and silent failure modes. ✓ indicates explicit coverage; ✗ indicates no reported coverage.

D ADDITIONAL RESULTS

D.1 EXTENDED WITHIN-BENCHMARK RESULTS

Table 2 in the main text reports the Qwen3-8B replication alongside the primary 4B target on the same 402-task Combined Retail held-out split. Table 12 gives mean±std over five evaluation seeds for the 4B rows; the 8B rows have no independent seed replication and are therefore omitted here.

The 8B results reproduce the qualitative complementarity between runtime BTM and RL observed for 4B (Table 2). Because the 8B checkpoint is non-thinking whereas the 4B target is a Thinking model, this replication varies both scale and reasoning mode and should not be interpreted as a controlled scaling comparison.

D.2 CROSS-BENCHMARK TRANSFER

The Retail-trained RL policy without target-domain BTM is +2.7pp above the base model on Airline (33.4%) and +1.5pp on BFCL (15.0%). Adding target-domain BTM raises the corresponding gains to +9.0pp and +5.0pp. On Telecom, where the 4B clean pass rate is only 21.1% (Table 1), gains are within noise.

Model	Train BTM	Infer BTM	Clean	Inj w/o Alt	Inj w/ Alt
4B	✗	✗	64.3±0.9	20.1±1.4	31.9±2.3
4B	✗	✓	65.3±0.9	36.9±1.2	43.6±1.6
4B + RL	✓	✗	64.5±0.8	26.4±1.2	38.8±2.2
4B + RL	✓	✓	63.9±1.3	**40.8**±1.6	**45.5**±3.0

Table 12: Qwen3-4B-Thinking-2507 results on the 402 held-out Combined Retail tasks over five evaluation seeds. Clean includes evaluation-rollout randomness; injected settings additionally include injection randomness. Corresponds to the 4B rows of Table 2.

Model	Train → Eval	Train BTM	Infer BTM	Inj w/o Alt
BFCL multi-turn (200 tasks)				
4B	— → BFCL	✗	✗	13.5
4B	— → BFCL	✗	✓	16.5
4B + RL	Retail → BFCL	✓	✗	15.0
4B + RL	Retail → BFCL	✓	✓	**18.5**
Airline-3I (584 tasks)				
4B	— → Airline	✗	✗	30.7
4B	— → Airline	✗	✓	36.6
4B + RL	Retail → Airline	✓	✗	33.4
4B + RL	Retail → Airline	✓	✓	**39.7**
τ^2-bench Telecom (114 tasks)				
4B	— → Telecom	✗	✗	19.3
4B	— → Telecom	✗	✓	21.1
4B + RL	Retail → Telecom	✓	✗	20.2
4B + RL	Retail → Telecom	✓	✓	**21.9**

Table 13: Cross-benchmark transfer for Qwen3-4B-Thinking-2507 under inject w/o alt. RL is trained only on Retail. The (✓,✗) rows evaluate the Retail-trained policy without target-domain BTM; the (✓,✓) rows additionally supply target-domain BTM. Values are point estimates with no seed-level variance.

D.3 Inference Cost

Table 14 reports inference cost for the main Qwen3-4B configurations on one evaluation run over the 402-task Combined Retail held-out set (seed 42).

	Clean		Inj w/o Alt		Inj w/ Alt	
Model	**Tokens**	**Steps**	**Tokens**	**Steps**	**Tokens**	**Steps**
Base	88K	11.9	70K	10.3	67K	9.9
Base + BTM	103K	12.2	110K	12.8	123K	12.5
RL + BTM	105K	12.3	117K	13.4	108K	11.7

Table 14: Aggregate token usage and mean episode length for the main Qwen3-4B configurations on the 402-task Combined Retail held-out set (seed 42). BTM-based configurations generally use more tokens under injection, consistent with additional recovery actions such as retries and alternative-tool calls.

E Qualitative Examples

We present two episodes illustrating retry and switching behaviors observed in the RL model. Each example contrasts the RL model with the base model on the same task; because injection seeds may differ due to hash randomization, these examples are illustrative rather than controlled paired comparisons.

Example 1: Retry (`task_78`). *Task:* User Yara Muller needs three actions—change address on order #W5056519, exchange an item in the same order, and cancel order #W5995614.

RL model (PASS, 10 tool calls): At step 2, `get_user_details` returns *HTTP 429 Too Many Requests*. The model immediately retries the same call at step 3, which succeeds. It then retrieves all three order details, modifies the address, exchanges the item, and cancels the third order, completing all sub-tasks.

Base model (FAIL, 9 tool calls): The base model successfully completes the first two sub-tasks. When `get_order_details` for the third order returns a schema validation error at step 8 ("Required field 'status' is missing"), it immediately calls `transfer_to_human_agents`, abandoning the final cancellation.

Key insight: In this trajectory, the RL policy retries after an explicit failure and succeeds, while the base model treats a later tool error as a global failure.

Example 2: Switch (`task_58`). *Task:* The user wants to exchange two items (coffee machine and laptop) in a delivered order.

RL model (PASS, 10 tool calls): After authentication and `get_user_details` succeed, the model calls `get_order_details`, which returns a schema-drift error ("Unknown field 'items_v2' "). It switches to `get_orders_for_user`, a fallback that provides order summaries, then uses `check_order_item_availability` and `get_product_details` to gather the remaining information and complete the exchange.

Base model (FAIL, 2 tool calls): The first call `find_user_id_by_name_zip` returns a malformed response. The base model immediately calls `transfer_to_human_agents` without attempting authentication via the alternative `find_user_id_by_email` path.

Key insight: In this trajectory, the RL policy switches to an alternative tool after a schema-drift error and recovers through an equivalent information path.