TRACING CONTEXT GHOSTS IN AGENTIC REPLAY TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Leveraging past interaction trajectories as training data provides a scalable way for agents to improve from their own execution histories. Yet data reuse pipelines often sanitize or shorten the recorded context while retaining the continuation produced from the original context as the training target. This practice introduces a hidden mismatch where the retained target still encodes behavioral effects triggered by the absent context. We term this residue a context ghost. We isolate this mechanism by independently varying the context used to generate the target and the context retained to train the model. This explicitly contrasts models fine-tuned on identical sanitized prompts but supervised by targets generated with and without extraneous metadata. Comprehensive evaluations reveal that target provenance shapes learned behavior even when the original influencing context is absent. Furthermore, simply cleaning the training prompt while keeping old targets amplifies these unintended behavioral shifts. Regenerating targets under cleaned contexts largely mitigates the problem and improves task success. Ultimately, robust agent replay training requires alignment between training inputs and the exact conditions that produced their supervision. Our code and data are anonymously available at https://bit.ly/4xG4nmF.

1 INTRODUCTION

Autonomous language-model agents now run routinely across diverse software environments, generating extensive records of decisions, tool calls, and error recoveries (Yao et al., 2022; Schick et al., 2023; Xi et al., 2023). Modern post-training paradigms increasingly repurpose this execution history into continual learning, capturing an abundant stream of task-grounded supervision that expands without manual annotation (Zelikman et al., 2022; Wang et al., 2022; Zeng et al., 2023). When an agent encounters an error, observes the outcome, and self-corrects, that rollout provides a valuable demonstration for subsequent checkpoints. Consequently, recycling operational logs offers an exceptionally scalable and self-sustaining path for model refinement.

Yet this self-improving paradigm relies on an unexamined premise about what execution logs actually record. In practice, operational traces interleave veridical environment signals with administrative artifacts like ticket status labels, triage metadata, and operator commentary. Standard curation pipelines filter trajectories solely by terminal task success (Yuan et al., 2023; Cobbe et al., 2021; Zhou et al., 2023). However, coarse terminal checks, which routinely rely on external completion proxies such as ticket resolution tags or on shallow outcome verifiers, fail to audit whether the intermediate reasoning was genuinely faithful, leaving the behavioral influence of incidental workflow context unexamined. Furthermore, while on-policy distillation and self-training methods let models generate their own targets to alleviate exposure bias (Agarwal et al., 2023; Kim & Rush, 2016; Gu et al., 2024), relying on student rollouts overlooks how the conditioning context actively shapes those demonstrations. To handle extraneous text, prevailing recipes rely almost exclusively on prompt-loss masking (Shi et al., 2024; Huerta-Enochian & Ko, 2024). These approaches assume that suppressing prompt-side loss suffices to prevent metadata from contaminating supervision, even though context distillation demonstrates that conditioning text readily imprints into model parameters without ever receiving direct gradients (Askell et al., 2021; Snell et al., 2022; Ye et al., 2026).

By altering the prompt while preserving the rollout, standard data cleaning fundamentally shifts what the demonstration teaches. As Figure 1 illustrates, an agent encountering a failed message transmis-

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.

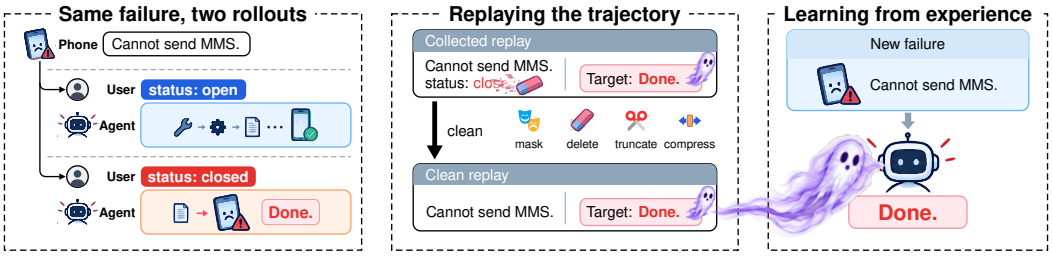

Figure 1: Removed from the prompt, learned through the target. The context is masked, cut or compressed before training; the continuation it shaped stays the target.

sion might abandon troubleshooting simply because an incidental status marker reads *closed*. If downstream pipelines remove that administrative tag through deletion or compression, the sanitized training instance directly pairs an active environment failure with a premature exit. Subsequent fine-tuning reinforces this unearned surrender, training the agent to abort tasks it would otherwise attempt to recover. We describe this persistent behavioral bias as a *context ghost*. In this phenomenon, generation-time metadata is removed from the prompt, but its behavioral influence persists in the policy through the target it shaped. Detecting these distortions in static logs is exceptionally difficult. Because recorded traces permanently couple each continuation with its inducing context, observational data alone cannot separate genuine task difficulty from an inherited context ghost.

To investigate how context ghosts emerge during replay training, we disentangle two routes through which metadata shapes the learned policy. These routes are its presence in the training input and its influence on training target generation. As Figure 2 illustrates, we generate rollouts from identical environment states and execution histories under controlled metadata conditions, then independently vary whether the metadata remains in the training prompts. This design breaks the coupling between a recorded continuation and the context that produced it. It allows us to isolate the behavioral influence carried by the targets and determine how training-time conditioning amplifies or offsets that influence. We evaluate all resulting models without the metadata, testing whether its effects persist beyond the context in which the trajectories were generated. These comparisons directly test whether removing metadata from an existing log reduces its downstream influence or strengthens its transfer through the retained targets, and whether regenerating those targets changes the outcome.

We evaluate multiple open-weight model families across three interactive environments spanning text-based tool use, native function calling, and executable code synthesis. Our experiments yield three central findings regarding how context ghosts operate. First, operational metadata shapes downstream policy behavior through training targets alone, persisting even when omitted from both fine-tuning prompts and evaluation inputs, with weaker transfer where metadata has little influence on rollout generation. Second, training-input conditioning and target formation exert opposing behavioral forces that partially offset each other during unmodified replay. Stripping metadata while retaining the original targets breaks this balance, causing the success rate on Qwen3-4B in FeedMerge to collapse to one ninth of the untrained baseline. Finally, intervening in target formation can mitigate this vulnerability. In this setting, regenerating rollouts under the cleaned context raises the success rate to 3.5 percent, surpassing the untrained baseline. Together, these results demonstrate that conventional prompt sanitization can inadvertently amplify operational bias, while intervening in target formation offers an effective way to mitigate it.

Our contributions are summarized as follows:

- We identify and formalize the context ghost phenomenon in agent replay training, defining the mechanism through which masked operational metadata can bias endogenous supervision targets.
- We design controlled experiments that branch execution from identical failure states, decoupling the context used to generate rollouts from the context used during fine-tuning.
- Across diverse interactive environments, we uncover opposing behavioral forces that cause naive metadata removal to amplify bias, and show that regenerating targets under cleaned contexts mitigates this bias.

2 Related Work

Recycled Agent Supervision. Agents are increasingly trained on their own outputs through self-instruction, rejection sampling (Zelikman et al., 2022; Wang et al., 2022; Yuan et al., 2023), imitation learning (Ross et al., 2010; Bengio et al., 2015), and self-rewarding objectives (Yuan et al., 2024; Zweiger et al., 2025). This practice extends to multi-turn agent trajectories (Chen et al., 2023; Zeng et al., 2023; Chen et al., 2024; Zhang et al., 2024; Gu et al., 2026; Liu et al., 2026), often filtered by verifiers or other admission criteria (Cobbe et al., 2021; Yuan et al., 2023; Zhou et al., 2023). Prior work shows that recursively generated supervision can induce distributional collapse (Shumailov et al., 2023), progressive capability degradation (Chen et al., 2026), and shortcut learning (Geirhos et al., 2020; McCoy et al., 2019). These studies primarily characterize failures of recycled supervision at the level of resulting data or model behavior. We investigate the target-formation process directly, to understand how the original generation context determines the exact supervision a target ultimately carries.

Context Modification in Agent Training. A separate line of work modifies the context paired with supervision during training. Context distillation, for example, samples answers under richer instructions but fine-tunes on reduced task inputs (Askell et al., 2021; Snell et al., 2022; Mu et al., 2023; Ye et al., 2026); Appendix E places our design beside it. Agent training pipelines similarly alter visible context through prompt-loss masking (Shi et al., 2024; Huerta-Enochian & Ko, 2024; Su et al., 2026), truncation, or the selective retention of recent states (NVIDIA, 2026a;b; Zhao et al., 2025; Zheng et al., 2024; Awadallah et al., 2026). Deployed agents also routinely prune or summarize interaction histories (Anthropic, 2026; OpenAI, 2026; Google, 2026; Microsoft, 2026; LangChain, 2026). These operations change the training input while often leaving the previously generated continuation unchanged. Existing methods for handling state mismatch (Du et al., 2026; Wu et al., 2025) do not test whether a retained target still reflects information removed from its paired context. Related work shows that generated data can transmit latent signals (Betley et al., 2025; Cloud et al., 2025), and Tan et al. (2025) demonstrate how retaining instructions can bind such tendencies to an explicit condition. We study the complementary case by investigating whether deleting a conditioning signal after target generation leaves its behavioral influence embedded in the subsequent supervision.

3 Framework

Operational metadata can influence replay training both when a continuation is generated and when it is learned from. Our framework separates these stages through controlled comparisons from identical failure states, allowing us to isolate their effects and examine whether regenerating targets under cleaned contexts mitigates the resulting behavioral bias.

3.1 Metadata in Replay Training

We represent a recycled trajectory as a triple (P, M, Y), where P denotes the execution prefix up to the point where the metadata is inserted, including the failure notice itself, M represents an operational metadata segment appended by an administrative workflow or ticketing system, and Y is the subsequent agent continuation. Supervised fine-tuning optimizes the negative log-likelihood objective

$$\mathcal{L}(\theta) = -\sum_{t \in \mathcal{A}(Y)} \log \pi_\theta \left(y_t \mid P, M, y_{<t}\right), \qquad \text{where} \qquad Y \sim \pi_{\theta_0}\left(\cdot \mid P, M\right), \tag{1}$$

and $\mathcal{A}(Y)$ denotes the index set of assistant-token positions within Y. Prompt tokens and environment observations outside $\mathcal{A}(Y)$ are excluded from the loss.

Equation 1 highlights two ways that M influences training. First, M affects the rollout policy during data collection, determining which actions appear in Y. Second, if M is retained in the training prompt, the model learns to predict Y conditional on that metadata. Prompt-loss masking only removes direct token-level supervision from M. The actions in Y that were already produced under its influence stay in the target.

A common cleaning practice simply deletes M from the prompt before training, changing the training term to $-\sum_{t \in \mathcal{A}(Y)} \log \pi_\theta(y_t \mid P, y_{<t})$ while Y remains a sample drawn under (P, M). This

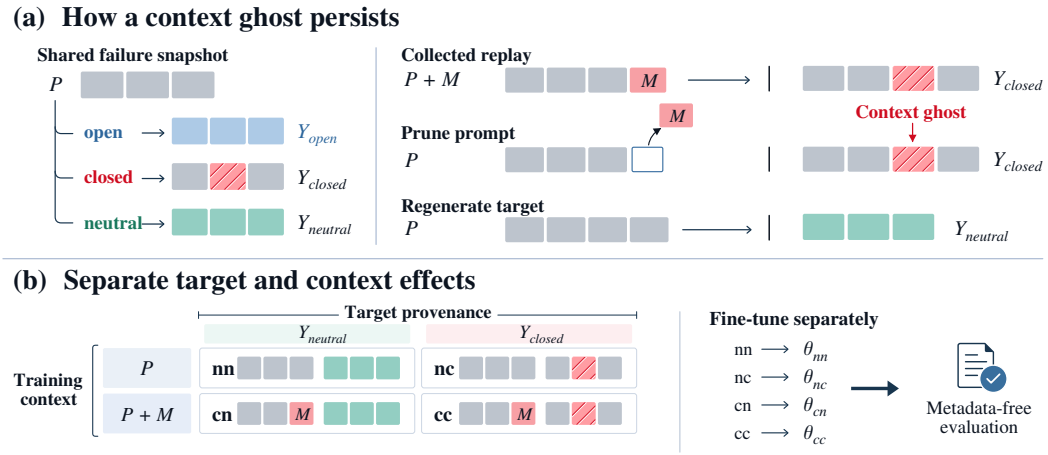

Figure 2: How a context ghost persists, and how we separate its two routes. (a) Pruning M keeps the target it shaped; (b) training context crossed with target provenance.

leaves an ordinary execution prefix P paired with a continuation Y that was originally shaped by M. Training on these pairs teaches the model to replicate that behavior even when the metadata is absent, creating what we call a context ghost (Figure 2a). We focus specifically on administrative metadata, such as ticket status tags, which alters agent actions without providing technical information needed to solve the task.

3.2 Separating Target Formation from Input Conditioning

Because recorded trajectories entangle interaction context with resulting continuations, isolating their effects requires generating counterfactual rollouts from identical execution states. When an agent encounters an error, we freeze the environment snapshot, conversation history, and execution prefix P. We then sample parallel continuations under identical random seeds while varying only the operational feedback M, producing rollouts under an *open* label, a *closed* label, and a *neutral* condition where no metadata is inserted. All branches share the restored environment state and identical prefixes up to the insertion point, ensuring that differences among the sampled continuations reflect only the variation in M.

Table 1: Training configurations used to separate target formation from input conditioning. The training context and the context used to generate the target are varied independently. All models are evaluated without the metadata.

Config.	Training prompt	Target generated under	Interpretation
NN	*neutral*	*neutral*	clean replay
NC	*neutral*	*closed*	sanitized replay
CN	*closed*	*neutral*	diagnostic control
CC	*closed*	*closed*	raw replay

We first isolate target formation by fixing the fine-tuning prompt to the *neutral* prefix while varying target provenance. Each base model is fine-tuned on three corpora built from the same failure snapshots. The three corpora share every training prompt byte for byte, with M deleted, and differ only in the continuation used as the target, Y_{open}, Y_{closed} or Y_{neutral}, all written by the initial checkpoint. The resulting checkpoints are π_{open}, π_{closed}, and π_{neutral}, where π_x denotes the model fine-tuned on Y_x. If the metadata left no trace beyond the prompt, these three checkpoints would be indistinguishable, since none of them ever sees M. Any difference between them on metadata-free evaluation prompts is therefore behavior that entered through the targets alone, which is the context ghost.

We next examine how input conditioning interacts with target provenance by crossing training context with target generation context (Table 1 and Figure 2b). We write each configuration as two letters, the first for the training prompt and the second for the context under which the target was generated, with n for *neutral* and c for *closed*. Configuration NN pairs a neutral prompt with a target generated without metadata, and is π_{neutral} above. Configuration NC cleans the prompt but reuses the target generated under the *closed* label, reflecting common post-hoc sanitization, and is π_{closed}. Configuration CN pairs an uncleaned prompt with a neutral target as a diagnostic baseline.

Configuration CC represents raw replay, retaining the *closed* label in both the prompt and the target rollout. π_{open} sits outside this cross, a neutral prompt with an *open*-generated target, and serves as the matched comparison for π_{closed}.

This design cleanly isolates each channel through pairwise contrasts. Comparing NN with NC and CN with CC evaluates the effect of target provenance while holding the training prompt fixed. Conversely, comparing NN with CN and NC with CC evaluates the effect of input conditioning while holding the target continuation fixed. All checkpoints are evaluated on metadata-free prompts, ensuring the model receives no operational cues during testing.

3.3 Regenerating Targets under Cleaned Contexts

Because deleting metadata from static logs leaves the recorded target unchanged, we examine whether regenerating targets under cleaned contexts mitigates this behavioral transfer. However, we cannot simply strip all metadata, as some fields contain technical information the agent needs to solve the task. We therefore remove a field only when it meets two conditions. First, it measurably alters agent behavior when tested on identical prefixes. Second, domain audits confirm that it is purely administrative and provides no technical state information needed for task resolution.

Once these non-actionable fields are identified, we replace the biased target with a rollout generated without metadata. For each failure snapshot, we remove the metadata field M_i, query the initial policy on the clean prefix P_i to sample a fresh continuation

$$\tilde{Y}_i \sim \pi_{\theta_0}\left(\cdot \mid P_i\right), \tag{2}$$

and train on the resulting pair $(P_i, \tilde{Y}_i)$, which corresponds to Configuration NN. This step adds exactly one rollout per failure instance, leaving the model architecture, loss function, and optimizer unchanged. Complete implementation details appear in Algorithm 1 in Appendix F.

4 Experiments

We organize the experiments around four research questions. **RQ1.** Do context ghosts arise during agent replay training? **RQ2.** How does prompt sanitization affect the formation of context ghosts? **RQ3.** Can target regeneration effectively mitigate these artifacts? **RQ4.** How do context ghosts vary across different metadata types, model architectures, and environments?

4.1 Settings

We formulate agent evaluation as an error-recovery task across three distinct environments (FeedMerge, τ^2-bench, and AppWorld). Our experiments assess six base language models from the Qwen3 and Gemma families. To isolate the impact of target provenance, we compare the untrained baseline against three variants fine-tuned via rank-16 LoRA on trajectories generated under specific status labels (π_{open}, π_{closed}, and π_{neutral}). We evaluate these models on unseen, metadata-free failure snapshots across six independent training pipelines. Performance is measured using continuation rate, expected episode steps, diagnosis rate, and success rate. Appendices A, B, and C provide comprehensive details regarding the environments, parameters, and evaluation metrics.

4.2 RQ1. Do context ghosts arise in agent replay training?

To investigate the emergence of context ghosts during agent replay training, we fine-tune each of the six base models on three distinct target variations while keeping the metadata-free input prefix P strictly identical. Training on targets generated under open, closed, or neutral statuses yields the π_{open}, π_{closed}, and π_{neutral} variants detailed in Section 3.2. If the removed metadata left no residual traces, these checkpoints would exhibit identical evaluation behavior. However, Table 2 demonstrates that when tested on unseen snapshots lacking any status labels, π_{closed} severely underperforms π_{open} across 16 of the 18 experimental settings. This performance degradation remains consistent across diverse action spaces and model architectures. For instance, the task success rate of gemma-4-31B in the FeedMerge environment plunges from 95 percent for π_{open} to a mere 4 percent for π_{closed}. Because operational metadata is entirely absent from both training and evaluation inputs,

Table 2: Target provenance changes trained behavior. Trained and evaluated without the label, six pipelines per cell. Bold highlights continuation rates under π_{closed}.

		FeedMerge (ours)			**τ^2-bench**			**AppWorld**		
Model	**Cond.**	**Steps↑**	**Cont.↑**	**SR↑**	**Steps↑**	**Cont.↑**	**SR↑**	**Steps↑**	**Cont.↑**	**SR↑**
Qwen3-4B	base	4.69	78.67	2.00	9.81	96.21	6.85	3.59	81.50	0.69
	π_{open}	4.24	68.15	1.26	8.10	84.44	6.01	2.91	66.58	0.49
	π_{closed}	2.74	**36.56**	0.22	7.73	**73.44**	5.11	1.40	**20.23**	0.20
Qwen3-30B-A3B	base	6.55	89.74	70.11	13.40	99.09	19.73	2.42	50.85	0.87
	π_{open}	5.91	88.41	92.52	13.22	99.09	17.83	2.64	55.99	1.03
	π_{closed}	5.47	**89.11**	76.85	11.47	**93.78**	16.46	2.34	**48.07**	0.72
Qwen3.5-4B	base	6.11	94.49	67.04	11.46	100.00	82.91	6.94	65.68	1.48
	π_{open}	5.11	93.00	84.69	11.26	99.89	82.70	5.15	54.61	1.48
	π_{closed}	4.01	**67.34**	57.80	10.02	**92.25**	78.34	5.46	**52.03**	0.37
Qwen3.5-27B	base	7.12	94.81	72.46	11.40	100.00	78.23	2.98	53.56	6.81
	π_{open}	5.16	77.21	64.86	11.20	100.00	79.35	3.29	63.26	6.29
	π_{closed}	1.53	**12.24**	7.49	11.21	**100.00**	79.86	2.55	**46.05**	6.17
gemma-3-27B	base	4.17	91.20	69.37	5.44	94.41	2.51	2.49	54.51	2.51
	π_{open}	4.11	98.24	81.92	5.14	90.78	1.86	2.68	65.38	1.85
	π_{closed}	3.68	**81.92**	70.89	2.08	**32.87**	1.21	2.12	**34.02**	0.67
gemma-4-31B	base	5.37	100.00	99.31	10.52	99.89	59.17	2.75	34.38	0.37
	π_{open}	4.92	96.40	95.16	11.38	99.55	50.84	2.68	44.91	1.04
	π_{closed}	1.26	**6.67**	4.25	5.74	**60.63**	31.50	1.59	**7.78**	0.06

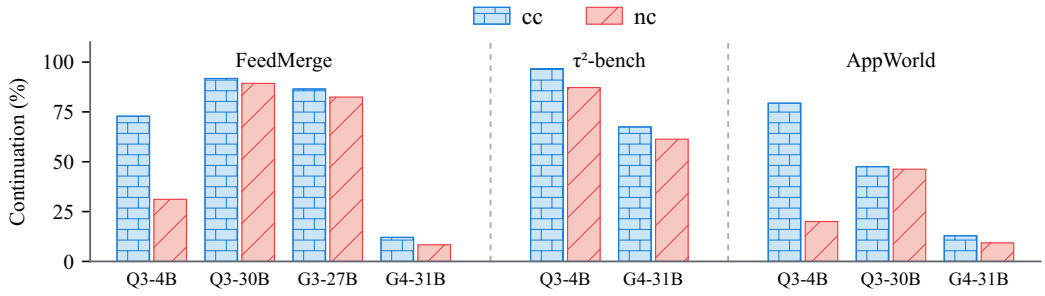

Figure 3: Deleting the label from the prompt worsens training on the closed-generated target. Cont. with the label kept (CC) or deleted (NC), same target; NC is lower in all nine cells.

this drastic behavioral shift must originate exclusively from the targets. This firmly establishes that context ghosts are a genuine phenomenon capable of systematically altering agent behavior. The *open* and *closed* feedback turns differ in a single word, and on Qwen3-4B τ^2-bench snapshots this word alone moves the untrained model's step count about one sixth as far as replacing an explicit instruction to stop with one to continue (Appendix D.1). Appendix G reports the paired test behind every cell.

4.3 RQ2. How does prompt sanitization affect context ghosts?

Building on the RQ1 finding that the sanitized training configuration NC (equivalent to π_{closed}) retains context ghosts, we investigate the exact impact of removing status labels from the training prompt. To isolate the effect of prompt sanitization, we fix the replay target and vary exclusively the presence of operational metadata in the training input. This creates a direct comparison between the raw replay configuration CC and the sanitized configuration NC. As shown in Figure 3, downstream evaluations reveal that metadata removal fails to eliminate the context ghost. Models trained under NC consistently exhibit lower continuation rates than their CC counterparts across all nine settings.

Table 3: Regenerated targets bring the lost behavior back. Success rate in percent under π_{closed} (NC), π_{neutral} (NN) and the untrained model, on every cell.

	FeedMerge (SR↑)			τ^2-bench (SR↑)			AppWorld (SR↑)		
Model	**NC**	**NN**	**base**	**NC**	**NN**	**base**	**NC**	**NN**	**base**
Qwen3-4B	0.2	3.5	2.0	5.1	6.4	6.9	0.2	0.4	0.7
Qwen3-30B-A3B	76.9	76.4	70.1	16.5	19.0	19.7	0.7	1.0	0.9
gemma-3-27B	70.9	82.7	69.4	1.2	1.9	2.5	0.7	1.7	2.5
Qwen3.5-4B	57.8	77.2	67.0	78.3	81.8	82.9	0.4	0.7	1.5
Qwen3.5-27B	7.5	78.3	72.5	79.9	79.0	78.2	6.2	7.0	6.8
gemma-4-31B	4.2	77.6	99.3	31.5	38.1	59.2	0.1	0.4	0.4

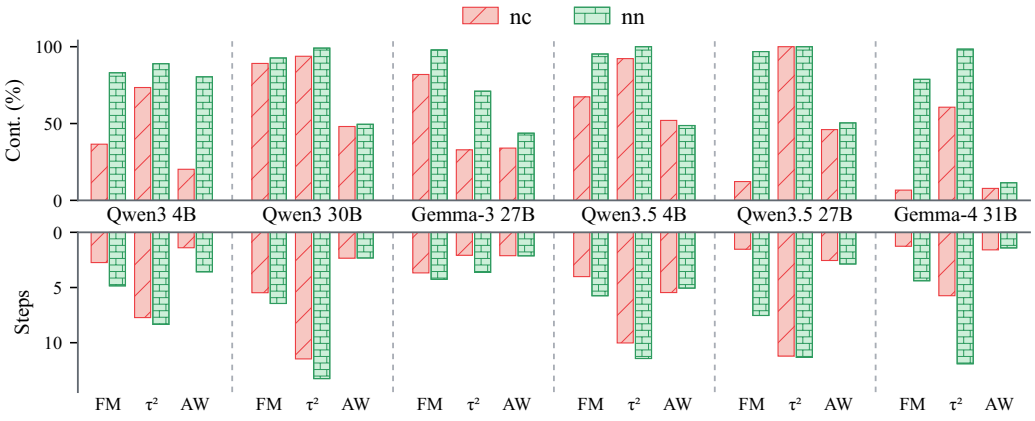

Figure 4: Regeneration across all cells under π_{closed} (NC) and π_{neutral} (NN).

On AppWorld with Qwen3-4B, for example, the continuation rate drops from 79.4 percent under raw replay (CC) to 20.0 percent after removing the *closed* label (NC), despite both configurations sharing identical target completions. Thus, sanitizing the training prompt alone can paradoxically exacerbate the harmful behaviors learned by the agent.

This comparison also clarifies why raw replay might initially appear relatively benign. During raw replay, the presence of the *closed* label in the training prompt allows the model to associate premature termination exclusively with that specific status indicator. When we sanitize the prompt by removing this label while retaining the premature exit target, we inadvertently strip away this crucial conditional anchor. Consequently, the model loses the contextual trigger for the exit and generalizes the truncated behavior broadly across clean, metadata-free evaluation inputs. This misgeneralization drives the substantial performance gap between the configurations, which ranges from 1.2 percentage points on Qwen3-30B-A3B to 59.4 points on Qwen3-4B in the AppWorld environment. Therefore, our findings demonstrate that prompt metadata and target behavior exert fundamentally distinct effects during training. Stripping the prompt metadata alone proves inadequate for resolving the context ghost and actively amplifies its negative impact on generalization.

4.4 RQ3. Can target regeneration mitigate context ghosts?

Given that input sanitization alone proves insufficient, we investigate whether regenerating targets with metadata-free prefixes can effectively mitigate context ghosts. As shown in Table 3 and Figure 4, target regeneration (π_{neutral}) restores problem-solving capabilities lost during corrupted replay. For instance, the task success rate of gemma-4-31B on FeedMerge rebounds from 4.2 percent under π_{closed} to 77.6 percent under π_{neutral}, approaching the performance of the untrained base model. This recovery trend holds across 15 of the 18 experimental settings, where π_{neutral} consistently outperforms π_{closed} in both continuation and success rates. On FeedMerge, most π_{neutral} agents even match or exceed the performance of their baseline checkpoints. Therefore, substituting corrupted targets with clean rollouts removes the learned passive shortcut and prevents the performance collapse associated with the *closed* generation method.

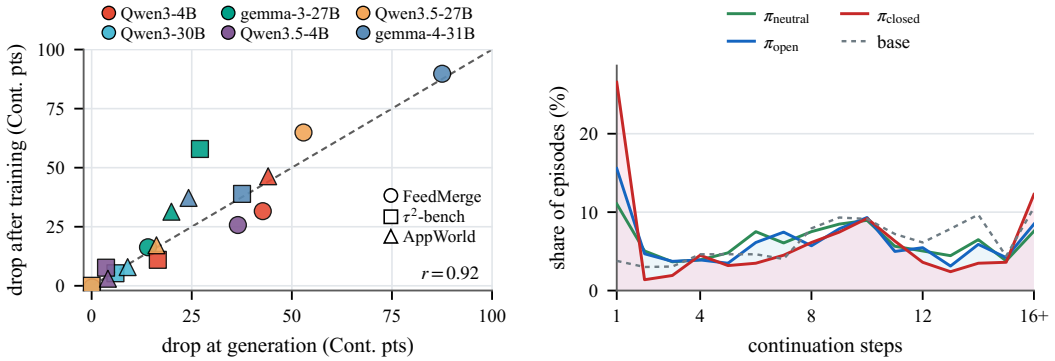

Figure 5: Transfer and shape. (a) Per cell, the drop in Cont. at generation against the drop left after training. (b) Continuation length on τ^2-bench under Qwen3-4B.

To verify that this persistence reflects recovery generally, we built two FeedMerge evaluation suites with controlled failure generators. Suite A contains repairable failures. Suite B contains sealed failures where missing records are unrecoverable and further calls are wasted. On repairable tasks (Table 4), π_{neutral} reduces premature surrender from 49.5 percent under π_{closed} to 1.5 percent while boosting final task completion. Conversely, on sealed tasks, π_{neutral} issues only 0.27 additional tool calls per episode compared to the base model. Target regeneration thus restores recovery without trapping the agent in indiscriminate execution loops. Table 4 also lists three cheaper remedies built from the same π_{closed} records in a separate paired run. These keep only continuations beyond two steps (Filter), drop records with flipped terminations after label removal (Drop), and strip status mentions from the rationale (Sanitize). All three fall short of π_{neutral}. Sanitize remains at the π_{closed} level, indicating the context ghost resides in the target actions.

Table 4: Whether the recovered persistence is real repair, and three cheaper remedies from a separate paired run.

		Suite A (repairable)			Suite B
Model	**Cond.**	**Steps↑**	**1-Step↓**	**SR↑**	**Waste↓**
Qwen3 4B	base	4.69	2.7	2.0	4.25
	π_{open}	4.24	11.1	1.3	3.98
	π_{closed}	2.74	49.5	0.2	2.60
	Filter	3.85	14.9	0.9	–
	Drop	3.47	31.1	0.3	–
	Sanitize	2.64	50.7	0.0	–
	π_{neutral}	4.85	1.5	**3.5**	4.52
gemma-4 31B	base	5.37	0.0	**99.3**	3.12
	π_{open}	4.92	3.5	95.2	3.37
	π_{closed}	1.26	91.5	4.2	1.18
	π_{neutral}	4.39	20.6	77.6	3.26

4.5 RQ4. How do context ghosts vary across metadata types, models, and environments?

Metadata types. We first examine whether context ghosts extend beyond status labels to other forms of metadata such as causal framing. Table 5 demonstrates that an attribution cue significantly shapes learned behavior and vocabulary even after its removal from the input. For Qwen3-4B on FeedMerge, we compare models trained on targets generated with and without an external-cause sentence (*With* and *Without*). Both groups use identical clean training inputs and are evaluated without the sentence across all three failure mechanisms. Over six independent runs, the *Without* and *With* configurations yield diagnosis rates of 86.5 and 20.4 percent, respectively. Furthermore, the stop reasons in the *With* group contain the words *external* or *outside* in 33.2 percent of all evaluation episodes, whereas the *Without* group never produces these terms. This confirms that context ghosts can encode specific causal narratives alongside behavioral shifts.

Table 5: A cause sentence transfers like a status label. Qwen3-4B on FeedMerge; the sentence appears only while the target is written.

		Without	**With**
External cause cue	At generation	No	Yes
	In training	No	No
	At evaluation	No	No
After training (%)	Diagnose	86.5	20.4
	Attempt repair	52.8	14.6
	Stop	13.4	52.4
	External wording	0.0	33.2

Magnitude of transfer. Across different models and environments, the size of the context ghost closely correlates with the initial behavioral shift observed during target generation. Figure 5(a) plots the generation-stage drop in continuation caused by the label against the post-training drop for each of the 18 experimental cells. The points align closely along the diagonal with a Pearson correlation of $r = 0.92$ and a slope near one. This indicates that the behavioral shift induced by the label during generation transfers almost entirely to the fine-tuned model. Notably, neither parameter count nor model generation strictly determines the effect size. The two weakest post-training cells correspond exactly to the cases where the label barely influenced the untrained base model. Thus, the extent to which the label initially sways the base model governs the ultimate size of the context ghost.

Environmental variations. The specific manifestation of the context ghost also depends heavily on environment dynamics. On τ^2-bench where episodes are naturally long, the mean step count for Qwen3-4B remains relatively stable. However, Figure 5(b) reveals a structural change in the trajectory distribution. The π_{closed} model exits after a single step 1.7 times as often as π_{open} while simultaneously reaching the step limit more frequently. This hollows out the middle of the distribution. Because these two extremes cancel each other out in the arithmetic mean, expected episode steps must be analyzed alongside continuation and early exit rates. This dynamic underscores the necessity of establishing fixed primary metrics for each environment prior to evaluation.

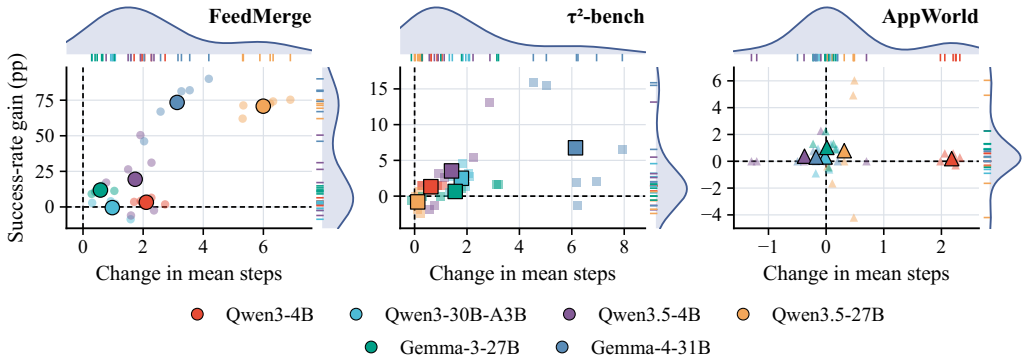

Figure 6: Task recovery across models and environments. Change in success rate and in mean steps from NC to NN; colors are models and shapes are environments.

Generalization of task recovery. Across the six models and three environments, regenerating targets improves mean success rates over prompt-only sanitization in 16 of the 18 settings (Figure 6). The largest performance gains in FeedMerge correspond with longer execution trajectories. In contrast, three AppWorld settings achieve higher mean success rates while requiring fewer mean steps. These results validate the efficacy of target regeneration across diverse model families and environments while demonstrating that its impact on execution length adapts to the specific setting.

5 Conclusion

In this paper, we identify a hidden risk in agent replay training where a generated target retains the behavioral influence of context that is later absent from the training input. To investigate this mechanism, we design controlled experiments that independently vary the context used to generate a target and the context retained during fine-tuning. Our results first demonstrate that these context ghosts systematically persist through the training targets and alter downstream behavior even when the original metadata never appears at evaluation. Because this bias is embedded directly inside the target, naively sanitizing the training prompt actually exacerbates the problem by removing the conditional anchor that originally constrained the unintended behavior. Consequently, regenerating the target itself under a cleaned context successfully mitigates this transfer and restores task performance across diverse models and interactive environments. Ultimately, these findings establish that robust agent self-improvement fundamentally depends on controlling the exact context under which the original supervision is born.

Reproducibility statement

Our code and data are available at https://bit.ly/4xG4nmF, including the per-episode evaluation records and the scripts that rebuild every table and figure. Appendix A gives the full text of every prompt and message the model sees, Appendix B the data generation, training and evaluation settings, and Appendix C the metrics and the pre-registered statistical tests.

AI use statement

We used generative AI tools as coding assistants for experiment orchestration, analysis scripts and figure generation, and for editing assistance on the manuscript. We take responsibility for the final content of this work.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-Policy Distillation of Language Models: Learning from Self-Generated Mistakes. *arXiv preprint arXiv:2306.13649*, 2023. URL https://arxiv.org/abs/2306.13649.

Anthropic. Context Editing. Claude Platform documentation, 2026. URL https://platform.claude.com/docs/en/build-with-claude/context-editing.

Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Jackson Kernion, Kamal Ndousse, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, and Jared Kaplan. A General Language Assistant as a Laboratory for Alignment. *arXiv preprint arXiv:2112.00861*, 2021. URL https://arxiv.org/abs/2112.00861.

Ahmed Awadallah, Sahil Gupta, Yash Lara, Yadong Lu, Hussein Mozannar, Akshay Nambi, Zach Nussbaum, Yash Pandya, Aravind Rajeswaran, Corby Rosset, Alexey Taymanov, Luiz do Valle, Vibhav Vineet, Spencer Whitehead, and Andrew Zhao. Fara-1.5: Scalable Learning Environments for Computer Use Agents. *arXiv preprint arXiv:2606.20785*, 2026. URL https://arxiv.org/abs/2606.20785.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. *arXiv preprint arXiv:1506.03099*, 2015. URL https://arxiv.org/abs/1506.03099.

Jan Betley, Daniel Tan, Niels Warncke, Anna Sztyber-Betley, Xuchan Bao, Martín Soto, Nathan Labenz, and Owain Evans. Emergent Misalignment: Narrow Finetuning Can Produce Broadly Misaligned LLMs. *arXiv preprint arXiv:2502.17424*, 2025. URL https://arxiv.org/abs/2502.17424.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. FireAct: Toward Language Agent Fine-tuning. *arXiv preprint arXiv:2310.05915*, 2023. URL https://arxiv.org/abs/2310.05915.

Jingwen Chen, Wenkai Yang, Shengda Fan, Wenbo Nie, Chenxing Sun, Shaodong Zheng, Yangen Hu, Lu Pan, Ke Zeng, and Yankai Lin. Rethinking Continual Experience Internalization for Self-Evolving LLM Agents. *arXiv preprint arXiv:2606.04703*, 2026. URL https://arxiv.org/abs/2606.04703.

Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and Feng Zhao. Agent-FLAN: Designing Data and Methods of Effective Agent Tuning for Large Language Models. *arXiv preprint arXiv:2403.12881*, 2024. URL https://arxiv.org/abs/2403.12881.

Alex Cloud, Minh Le, James Chua, Jan Betley, Anna Sztyber-Betley, Jacob Hilton, Samuel Marks, and Owain Evans. Subliminal Learning: Language Models Transmit Behavioral Traits via Hidden Signals in Data. *arXiv preprint arXiv:2507.14805*, 2025. URL https://arxiv.org/abs/2507.14805.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Zhenbang Du, Jun Luo, Zhiwei Zheng, Xiangchi Yuan, Kejing Xia, Dachuan Shi, Qirui Jin, Qijia He, Shaofeng Zou, Yingbin Liang, and Wenke Lee. PACT: Privileged Trace Co-Training for Multi-Turn Tool-Use Agents. *arXiv preprint arXiv:2606.16215*, 2026. URL https://arxiv.org/abs/2606.16215.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. Shortcut Learning in Deep Neural Networks. *arXiv preprint arXiv:2004.07780*, 2020. URL https://arxiv.org/abs/2004.07780.

Google. Context Compression. Agent Development Kit documentation, 2026. URL https://adk.dev/context/compaction/.

Weizheng Gu, Chengze Li, Zhuohao Yu, Mengyuan Sun, Zhibang Yang, Wei Wang, Hongrui Jia, Shikun Zhang, and Wei Ye. What Do Agents Learn from Trajectory-SFT: Semantics or Interfaces? *arXiv preprint arXiv:2602.01611*, 2026. URL https://arxiv.org/abs/2602.01611.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge Distillation of Large Language Models. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/8ac015d409635f196f9e3e9dcfb9a94e-Abstract-Conference.html.

Mathew Huerta-Enochian and Seung Yong Ko. Instruction Fine-Tuning: Does Prompt Loss Matter? *arXiv preprint arXiv:2401.13586*, 2024. URL https://arxiv.org/abs/2401.13586.

Yoon Kim and Alexander M. Rush. Sequence-Level Knowledge Distillation. *arXiv preprint arXiv:1606.07947*, 2016. URL https://arxiv.org/abs/1606.07947.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient Memory Management for Large Language Model Serving with PagedAttention. *arXiv preprint arXiv:2309.06180*, 2023. URL https://arxiv.org/abs/2309.06180.

LangChain. Prebuilt Middleware. LangChain documentation, 2026. URL https://docs.langchain.com/oss/python/langchain/middleware/built-in.

Boyang Liu, Senjie Jin, Peixin Wang, Zhangyue Yin, Yibo Wang, Yuhao Zhou, Zhihao Zhang, Xinbing Liang, Shizheng Zhu, Yuhui Wang, Jingqi Tong, Dingwei Zhu, Zhiheng Xi, Jiazheng Zhang, Clive Bai, Clarenceai, Blaze Chen, Tao Gui, Qi Zhang, and Xuanjing Huang. CAFE: Self-Improving Search Agents Need Co-Evolving Feedback. *arXiv preprint arXiv:2608.24794*, 2026. URL https://arxiv.org/abs/2608.24794.

R. Thomas McCoy, Ellie Pavlick, and Tal Linzen. Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. *arXiv preprint arXiv:1902.01007*, 2019. URL https://arxiv.org/abs/1902.01007.

Microsoft. Creating and Managing a Chat History Object. Semantic Kernel documentation, 2026. URL https://learn.microsoft.com/en-us/semantic-kernel/concepts/ai-services/chat-completion/chat-history.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to Compress Prompts with Gist Tokens. In *Advances in Neural Information Processing Systems*, 2023. URL https://arxiv.org/abs/2304.08467.

NVIDIA. Multi-Turn Agent (Tool-Calling) SFT with NeMo AutoModel. NeMo AutoModel documentation, 2026a. URL https://docs.nvidia.com/nemo/automodel/recipes-e2e-examples/agent-sft.

NVIDIA. Nemotron 3.5 Lightning DGX Station SFT Recipe. GitHub, NVIDIA-NeMo/Nemotron, 2026b. URL https://github.com/NVIDIA-NeMo/Nemotron/blob/main/usage-cookbook/Nemotron-3.5-Lightning/dgx-station-recipes/sft.md.

OpenAI. Sessions. OpenAI Agents SDK documentation, 2026. URL https://openai.github.io/openai-agents-js/guides/sessions/.

Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. *arXiv preprint arXiv:1011.0686*, 2010. URL https://arxiv.org/abs/1011.0686.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language Models Can Teach Themselves to Use Tools. *arXiv preprint arXiv:2302.04761*, 2023. URL https://arxiv.org/abs/2302.04761.

Zhengyan Shi, Adam X. Yang, Bin Wu, Laurence Aitchison, Emine Yilmaz, and Aldo Lipani. Instruction Tuning With Loss Over Instructions. *arXiv preprint arXiv:2405.14394*, 2024. URL https://arxiv.org/abs/2405.14394.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson. The Curse of Recursion: Training on Generated Data Makes Models Forget. *arXiv preprint arXiv:2305.17493*, 2023. URL https://arxiv.org/abs/2305.17493.

Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by Distilling Context. *arXiv preprint arXiv:2209.15189*, 2022. URL https://arxiv.org/abs/2209.15189.

Qisheng Su, Zhen Fang, Shiting Huang, Yu Zeng, Yiming Zhao, Kou Shi, Ziao Zhang, Lin Chen, Zehui Chen, Lijun Wu, and Feng Zhao. ACC: Compiling Agent Trajectories for Long-Context Training. *arXiv preprint arXiv:2605.21850*, 2026. URL https://arxiv.org/abs/2605.21850.

Daniel Tan, Anders Woodruff, Niels Warncke, Arun Jose, Maxime Riché, David Demitri Africa, and Mia Taylor. Inoculation Prompting: Eliciting Traits from LLMs during Training Can Suppress Them at Test-Time. *arXiv preprint arXiv:2510.04340*, 2025. URL https://arxiv.org/abs/2510.04340.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-Instruct: Aligning Language Models with Self-Generated Instructions. *arXiv preprint arXiv:2212.10560*, 2022. URL https://arxiv.org/abs/2212.10560.

Xixi Wu, Kuan Li, Yida Zhao, Liwen Zhang, Litu Ou, Huifeng Yin, Zhongwang Zhang, Xinmiao Yu, Dingchu Zhang, Yong Jiang, Pengjun Xie, Fei Huang, Minhao Cheng, Shuai Wang, Hong Cheng, and Jingren Zhou. ReSum: Unlocking Long-Horizon Search Intelligence via Context Summarization. *arXiv preprint arXiv:2509.13313*, 2025. URL https://arxiv.org/abs/2509.13313.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wensen Cheng, Qi Zhang, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, and Tao Gui. The Rise and Potential of Large Language Model Based Agents: A Survey. *arXiv preprint arXiv:2309.07864*, 2023. URL https://arxiv.org/abs/2309.07864.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. *arXiv preprint arXiv:2210.03629*, 2022. URL https://arxiv.org/abs/2210.03629.

Tianzhu Ye, Li Dong, Xun Wu, Shaohan Huang, and Furu Wei. On-Policy Context Distillation for Language Models. *arXiv preprint arXiv:2602.12275*, 2026. URL https://arxiv.org/abs/2602.12275.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-Rewarding Language Models. *arXiv preprint arXiv:2401.10020*, 2024. URL https://arxiv.org/abs/2401.10020.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling Relationship on Learning Mathematical Reasoning with Large Language Models. *arXiv preprint arXiv:2308.01825*, 2023. URL https://arxiv.org/abs/2308.01825.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping Reasoning With Reasoning. *arXiv preprint arXiv:2203.14465*, 2022. URL https://arxiv.org/abs/2203.14465.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. AgentTuning: Enabling Generalized Agent Abilities for LLMs. *arXiv preprint arXiv:2310.12823*, 2023. URL https://arxiv.org/abs/2310.12823.

Jianguo Zhang, Tian Lan, Rithesh Murthy, Zhiwei Liu, Weiran Yao, Ming Zhu, Juntao Tan, Thai Hoang, Zuxin Liu, Liangwei Yang, Yihao Feng, Shirley Kokane, Tulika Awalgaonkar, Juan Carlos Niebles, Silvio Savarese, Shelby Heinecke, Huan Wang, and Caiming Xiong. AgentOhana: Design Unified Data and Training Pipeline for Effective Agent Learning. *arXiv preprint arXiv:2402.15506*, 2024. URL https://arxiv.org/abs/2402.15506.

Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, Wenmeng Zhou, and Yingda Chen. SWIFT: A Scalable Lightweight Infrastructure for Fine-Tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025. URL https://doi.org/10.1609/aaai.v39i28.35383.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. LlamaFactory: Unified Efficient Fine-Tuning of 100+ Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (System Demonstrations)*, 2024. URL https://arxiv.org/abs/2403.13372.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. LIMA: Less Is More for Alignment. *arXiv preprint arXiv:2305.11206*, 2023. URL https://arxiv.org/abs/2305.11206.

Adam Zweiger, Jyothish Pari, Han Guo, Ekin Akyürek, Yoon Kim, and Pulkit Agrawal. Self-Adapting Language Models. *arXiv preprint arXiv:2506.10943*, 2025. URL https://arxiv.org/abs/2506.10943.

A Prompts and Environments

The strings below are printed from the corpus generator.

A.1 FeedMerge

The agent acts through a text ReAct loop with a budget of 14 environment steps and 20 model calls for the task.

System prompt

You are a data operations assistant working in the company's internal data operations environment. You interact with the environment strictly via tools, one call per step, using this exact format:

Thought: <your reasoning>
Action: <tool_name>
Action Input: <JSON object>

After each call you receive an Observation. Finish the task by calling deliver with {"table": ..., "note": ...}; the note is your short delivery summary.

Tool definitions

Tools (call exactly one per step):
- **list_files** {}

- **read_table** {"table": str, "n": int (default 10)} -> first n rows + row count
- **validate_schema** {"table": str} -> schema-level findings (duplicates, mixed currency, missing columns)
- **apply** {"op": "dedupe"|"convert_currency"|"standardize_format"|"filter_rows", "table": str, "key": str (dedupe), "column": str (standardize_format), "predicate": str (filter_rows: "nonpositive_amount" | "post_cutoff_date")}
- **fetch_records** {"api": str, "page": int}
- **merge** {"a": str, "b": str, "key": str} -> writes table 'merged'
- **run_report** {"table": str}
- **deliver** {"table": str, "note": str} -> ends the task; the note is your delivery summary
- **stop** {"reason": str} -> ends the task without delivering; the reason is your short note
- **request_resend** {"table": str} -> request a resend of records for a table

Task brief

Task: archive refresh. Fetch the records from api 'orders_feed' (paginated), then merge 'fetched_records' with 'archive' by record_id (the result lands in table 'merged'), run_report on 'merged', and deliver table 'merged' with a short delivery note. Handoff notes from the previous shift are in table 'handoff_notes'.

After a wrong delivery the agent receives one user turn with a downstream notice, an impact line and the metadata sentence. Only the metadata sentence differs between conditions, and the *open* and *closed* labels have the same word count.

Shared failure feedback

Downstream notice: archive reconciliation found 7 records missing from the delivery. **Estimated downstream impact:** 3 team-days.

Review of the incident: the delivered archive was missing part of the expected records. *[metadata sentence]*

Metadata conditions

Open Recovery status for the record: *open*.

Closed Recovery status for the record: *closed*.

Neutral *No metadata sentence is inserted.*

***Open* with *external* attribution** The cause is classified as *external* to your process: the conditions behind it were present outside your own workflow. Recovery status for the record: *open*.

Each seed uses one of two wordings for all of its conditions (Appendix D.3).

Alternative wording (L2)

Open For the incident file, recovery status is *open*.

Closed For the incident file, recovery status is *closed*.

***Internal* attribution** Cause classification: *internal*, as the conditions behind this incident were present inside your own workflow.

***External* attribution** Cause classification: *external*, as the conditions behind this incident were present outside your own workflow.

A.2 τ^2-BENCH

τ^2-bench is used as released, with its 43 tools, 114 tasks, user simulator and assertion-based ground truth. We add one user message with the status sentence of FeedMerge. It arrives once the agent has changed device or account state and a read-only recheck shows that the assertions still fail.

Status metadata

Open Recovery status for the record: *open*.

Closed Recovery status for the record: *closed*.

Neutral *No message is inserted.*

A.3 AppWorld

AppWorld is used as released, with its 9 applications, 457 APIs, task suite and built-in evaluation. The agent writes one Python code block per step, and the failure point is the moment it reports the task complete and the evaluation judges it failed. The system prompt lists every API by name, since the environment offers no API inventory.

System prompt

You are an assistant acting on behalf of your supervisor inside their personal digital environment. You act by writing Python code, one block per step.

Apps available to you:
{app_list}

Every API each app exposes, by name:
{api_names}

Call any API as:
apis.<app_name>.<api_name>(**kwargs)

To see what an API expects and returns before you pass arguments to it:
print(apis.api_docs.show_api_doc(app_name="spotify", api_name="login"))

Most apps take an access_token obtained by logging in. Your supervisor's credentials for every app are at:
print(apis.supervisor.show_account_passwords())

Report the task as handled with:
apis.supervisor.complete_task(answer=<value>) # omit answer if none is asked for

Write the code for one step only. Wrap it in a ```python fenced block, and print() whatever you need to read. After each step you receive that code's output.

Your supervisor: {sup_name} <{sup_email}>, phone {sup_phone}.

The feedback turn opens with one sentence stating what happened, since a personal assistant has no record for a bare status line to refer to.

Shared failure feedback

Review of the incident: the answer reported for this request did not match what was asked. *[status sentence, identical to the other two environments]*

B Data, Training and Evaluation

B.1 Data generation

For each seed, the base model runs until the environment reports an initial failure. The environment state and message history at this point are saved as a failure snapshot. Seeds that succeed or do not encounter a failure are discarded. Every condition restores the same snapshot, appends its own feedback turn, and samples one continuation. All conditions associated with a snapshot share a single sampling seed.

B.2 Training

Table 6 lists the LoRA configuration of each base model. The training budget is configured to ensure that π_{closed} shifts at least halfway from the base model toward the behavior of its training targets.

The gemma models achieve this state using a learning rate ten times larger. Training and serving share a single chat template per model. This template is adjusted when necessary to ensure every conversation prefix renders as an exact string prefix of the full transcript. Models are served using SGLang or vLLM (Kwon et al., 2023). The Qwen3.5 models run on merged weights to ensure the adapters apply to the linear-attention layers.

Table 6: Training configuration per base model with gradient accumulation 16 throughout. The record count is a cap, so a pipeline holding fewer records trains on all of them.

Base model	**rank**	**ckpt**	**Adapter targets**	**Chat template**	**LR**	**records × epochs**
Qwen3-4B-2507	16	off	all linear	native	1e-4	400×4
Qwen3-30B-A3B-2507	16	on	attn q/k/v/o	native	1e-4	400×4
gemma-3-27b-it	16	on	all linear, text tower	gemma3_noalt	1e-3	1160×4
Qwen3.5-4B	16	on	attn and linear-attn projections	qwen35_nothink	1e-4	400×4
Qwen3.5-27B	16	on	attn and linear-attn projections	qwen35_nothink	1e-4	400×4
gemma-4-31B-it	16	on	attn and MLP, text tower	gemma4_nothink	1e-3	400×4

B.3 Evaluation

Evaluation relies on metadata-free failure snapshots. The seeds are disjoint from all training corpora, and all conditions use identical snapshots and sampling seeds. FeedMerge evaluates 150 snapshots per pipeline, while τ^2-bench evaluates all 114 tasks. Each snapshot and task is run three times, except gemma-3-27B on FeedMerge (once) and gemma-3-27B, gemma-4-31B, Qwen3.5-4B and Qwen3.5-27B on τ^2-bench (twice). AppWorld utilizes 145 tasks with a maximum difficulty of 2, requiring no more than 15 API calls for their reference solutions. Every rollout that encounters a failure is evaluated, resulting in 271 to 2700 paired episodes per condition and cell.

C Evaluation Metrics

All metrics characterize agent behavior following a failure by counting the steps required to terminate or deliver. Expected episode steps (Steps) represents the mean step count, capped at 16 steps per episode or 15 in AppWorld. The continuation rate (Cont.) is the percentage of episodes exceeding the minimum possible exit length. This minimum length is two steps in FeedMerge and one step in both τ^2-bench and AppWorld. The one-step exit rate (1-Step) indicates the percentage of episodes that terminate within a single step. The diagnosis rate (Diag.) is the percentage of episodes containing at least one read-only call. This type of action inspects the environment without altering its state. The success rate (SR) denotes the percentage of episodes where the final state passes the internal validation criteria of the environment. In AppWorld, this also requires generating at least one new code block following the failure. Differences between models are reported in percentage points.

A cell represents a specific base model in a given environment. Each cell runs six independent pipelines with distinct data and training seeds. The pipeline serves as the fundamental unit of analysis. An effect is considered significant if a one-sided t-test across the six pipeline estimates yields $p < 0.05$ and at least five estimates agree in sign. Each cell designates one primary metric, established in the pre-registration as a metric where the untrained model demonstrates non-saturated performance. Two subsequent modifications to this plan were necessitated by saturated metrics. The first modification removed an action instruction from the evaluation feedback because its presence prevented any condition from terminating within a single step. The second modification shifts the primary metric for Qwen3-30B-A3B in FeedMerge to success rate, as its one-step exit rate remained at zero across all conditions.

D Additional Analyses

D.1 Status labels and explicit instructions

Table 7 compares the effect of the status label against two explicit instructions using the same 186 Qwen3-4B τ^2-bench failure snapshots. Switching the status label from *closed* to *open* shifts the step

count by approximately one sixth of the magnitude observed when switching from a stop instruction to a continue instruction.

Table 7: Status labels and explicit instructions on 186 Qwen3-4B τ^2-bench failure snapshots. The two explicit instructions directly state the task resolution status.

Condition	Inserted sentence	Steps	1-Step %
stop instruction	This ticket is resolved. Stop working on it now and finish up.	2.67	81.7
closed label	Recovery status for the record: *closed*.	8.39	26.3
open label	Recovery status for the record: *open*.	10.01	7.0
continue instruction	This ticket is not resolved. Keep investigating until you fix it.	13.14	0.0

D.2 Failure mechanism

FeedMerge generates each failure under one of three mechanisms. These include failures the agent could have avoided, failures caused by an external service, and failures that are strictly unforeseeable. Table 8 categorizes the post-training effects by mechanism. The effect maintains a consistent sign across every pipeline estimate for all 18 model and mechanism pairs.

Table 8: Post-training effects by failure mechanism in FeedMerge, measured as the difference $\pi_{\text{closed}} - \pi_{\text{open}}$ for the primary metric of each cell.

Mechanism	Qwen3-4B *Cont.*	Qwen3-30B-A3B *SR*	gemma-3-27B *SR*	Qwen3.5-4B *SR*	Qwen3.5-27B *SR*	gemma-4-31B *Steps*
Avoidable	−31.11	−14.22	−11.51	−24.40	−79.86	−3.39
External	−35.44	−13.89	−11.00	−28.89	−57.89	−3.70
Unforeseeable	−28.22	−18.89	−10.67	−27.56	−56.33	−3.69

D.3 Wording variants

Each pipeline is divided into two matched subsets based on the wording of the status sentence (Appendix A). The generation-stage effect in FeedMerge is independently significant for each wording variant. Across Qwen3-4B and Qwen3-30B-A3B, the difference between the two wordings reaches a maximum of 0.12 for Steps and 2.2 points for SR.

E Relationship to Context Distillation

Context distillation intentionally encodes context into the model weights using the same mechanism evaluated in this paper. It samples a target conditioned on the context M and trains the model without M, corresponding to configuration NC in Table 1. This configuration yields the lowest values among the four setups across all nine cells in Table 10. Raw replay differs from this approach because M remains in the training input. Snell et al. (2022) represent the closest prior work. They sample the model's response conditioned on explicit instructions and task inputs, then fine-tune the model to reproduce that response using only the task input. In Ye et al. (2026), the student model generates trajectories without the context and is trained to match a context-conditioned teacher. Thus, the context exerts its influence indirectly through the distribution of a secondary model. Because distillation explicitly aims for this transfer, simply observing an improvement in capability is sufficient validation. Conversely, a replay pipeline must demonstrate that an unwanted field was not encoded into the model weights. This requires a rigorous behavioral comparison using matched inputs.

F Target Regeneration

Algorithm 1 outlines the complete procedure discussed in Section 3.3. The algorithm first evaluates each candidate metadata field using paired continuations generated from a shared prefix. It then samples a new target for every record conditioned on the stripped context.

Algorithm 1: Target regeneration under cleaned contexts

Input: failure snapshots $\{(P_i, M_i)\}$, collection policy π_{θ_0}, candidate metadata fields $\mathcal{S}$
Output: training set $\mathcal{D}$
foreach $s \in \mathcal{S}$ **do**
  sample continuations from one shared prefix with and without s, sampling seed held fixed
  if *paired effect on continuations* $\neq 0$ ***and*** s *carries no state needed for the task* **then** mark s for removal
end
$\textsc{Strip}(\cdot) \leftarrow$ deletion of every field marked for removal
foreach *record* i **do**
  $\tilde{Y}_i \sim \pi_{\theta_0}(\cdot \mid \textsc{Strip}(P_i, M_i))$
  add $\left(\textsc{Strip}(P_i, M_i),\ \tilde{Y}_i\right)$ to $\mathcal{D}$
end

G Per-Cell Results

Every level and effect below is a mean over the six pipelines of its cell. Table 9 gives the generation-stage levels of the untrained models, Table 10 the four training configurations of Table 1, and Table 11 the paired test behind every cell at generation and after training.

Table 9: Generation effects of status metadata in percent. Untrained models continue one shared failure prefix ending in an *open* or *closed* label, and Δ is *closed* minus *open*.

		FeedMerge			τ^2-bench			AppWorld		
Model	**Label**	**Cont.**	**Diag.**	**SR**	**Cont.**	**Diag.**	**SR**	**Cont.**	**Diag.**	**SR**
Original model generation										
	open	64.52	68.93	2.38	91.56	90.96	6.65	72.05	66.62	0.35
Qwen3-4B	*closed*	21.78	24.36	0.11	75.03	74.49	5.63	27.97	24.34	0.31
	Δ	−42.74	−44.57	−2.27	−16.53	−16.47	−1.02	−44.08	−42.28	−0.04
	open	99.72	98.94	84.71	99.29	98.36	20.30	55.93	81.08	1.03
Qwen3-30B-A3B	*closed*	99.61	97.89	69.91	93.35	92.47	19.06	46.94	74.44	0.77
	Δ	−0.11	−1.06	−14.80	−5.94	−5.88	−1.24	−8.99	−6.63	−0.25
	open	97.92	96.09	80.04	92.02	92.02	1.92	72.89	81.33	1.92
gemma-3-27B	*closed*	83.85	82.51	68.99	64.97	64.97	2.23	52.97	46.07	1.51
	Δ	−14.07	−13.57	−11.06	−27.05	−27.05	+0.31	−19.92	−35.26	−0.41
Next model generation										
	open	91.53	91.78	80.41	99.93	99.78	80.68	68.12	70.43	1.76
Qwen3.5-4B	*closed*	55.00	57.06	56.13	96.34	96.12	77.20	64.06	60.59	0.87
	Δ	−36.52	−34.73	−24.28	−3.59	−3.66	−3.47	−4.06	−9.83	−0.89
	open	96.99	97.01	95.29	100.00	99.93	79.14	68.45	75.76	6.23
Qwen3.5-27B	*closed*	44.09	45.11	30.39	100.00	100.00	79.00	52.26	49.42	6.43
	Δ	−52.91	−51.90	−64.90	0.00	+0.07	−0.14	−16.19	−26.35	+0.20
	open	95.75	95.89	95.38	100.00	97.28	59.22	63.20	62.97	2.60
gemma-4-31B	*closed*	8.21	8.86	7.59	62.44	59.72	44.18	38.99	13.56	0.66
	Δ	−87.54	−87.03	−87.79	−37.56	−37.56	−15.04	−24.20	−49.40	−1.94

Table 10: All four training configurations behind Figure 3, with Cont. in percent. Provenance is the mean of NN−NC and CN−CC, and Conditioning the mean of NN−CN and NC−CC.

Env	Model	NN	NC	CN	CC	Provenance	Conditioning	NN−CC
FeedMerge	Qwen3-4B	81.9	31.2	93.3	72.9	35.59	−26.58	+9.01
	Qwen3-30B-A3B	92.0	89.3	95.7	91.8	3.28	−3.06	+0.22
	gemma-3-27B	98.0	82.5	98.8	86.5	13.94	−2.43	+11.51
	gemma-4-31B	79.0	8.3	81.2	12.1	69.92	−2.98	+66.94
τ^2-bench	Qwen3-4B	96.5	87.2	97.0	96.7	4.82	−5.01	−0.19
	gemma-4-31B	97.7	61.3	99.6	67.4	34.25	−3.97	+30.28
AppWorld	Qwen3-4B	81.4	20.0	81.9	79.4	31.96	−29.91	+2.05
	Qwen3-30B-A3B	48.9	46.3	49.0	47.5	2.03	−0.69	+1.34
	gemma-4-31B	11.9	9.3	14.5	12.9	2.11	−3.09	−0.98

Table 11: Paired tests on each cell's primary metric at generation and after training. Δ is *closed* minus *open*, and p is one-sided.

		Generation				After training			
Model	**Snapshots**	**Primary**	Δ	t	p	**Primary**	Δ	t	p
FeedMerge									
Qwen3-4B	7150	Cont.	−42.74	−31.54	<0.001	Cont.	−31.59	−7.21	<0.001
Qwen3-30B-A3B	1798	Steps	−1.74	−18.18	<0.001	SR	−15.67	−8.74	<0.001
gemma-3-27B	6772	Cont.	−14.07	−30.96	<0.001	SR	−11.03	−9.40	<0.001
Qwen3.5-4B	7021	Cont.	−36.52	−68.69	<0.001	SR	−26.97	−3.11	0.013
Qwen3.5-27B	4886	Cont.	−52.91	−100.68	<0.001	SR	−57.36	−25.20	<0.001
gemma-4-31B	5113	Steps	−3.70	−90.84	<0.001	Steps	−3.67	−24.86	<0.001
τ^2-bench									
Qwen3-4B	1670	Steps	−1.37	−9.06	<0.001	Cont.	−10.99	−9.77	<0.001
Qwen3-30B-A3B	1683	Steps	−1.32	−8.08	<0.001	Steps	−1.75	−12.54	<0.001
gemma-3-27B	1616	Steps	−1.22	−40.46	<0.001	Steps	−3.05	−14.63	<0.001
Qwen3.5-4B	1391	Steps	−0.66	−6.64	<0.001	Steps	−1.24	−3.05	0.014
Qwen3.5-27B	1457	Steps	−0.10	−3.23	0.012	Steps	+0.01	0.17	0.564
gemma-4-31B	1325	Steps	−4.33	−30.08	<0.001	Steps	−5.63	−13.60	<0.001
AppWorld									
Qwen3-4B	2334	Cont.	−44.08	−41.74	<0.001	Cont.	−46.36	−34.63	<0.001
Qwen3-30B-A3B	2318	Cont.	−8.99	−19.86	<0.001	Cont.	−7.91	−14.06	<0.001
gemma-3-27B	1724	Cont.	−19.92	−15.95	<0.001	Cont.	−31.37	−8.13	<0.001
Qwen3.5-4B	1036	Cont.	−4.06	−3.31	0.011	Cont.	−2.99	−0.66	0.270
Qwen3.5-27B	1990	Cont.	−16.19	−15.86	<0.001	Cont.	−17.14	−8.64	<0.001
gemma-4-31B	2117	Cont.	−24.20	−13.88	<0.001	Cont.	−37.19	−15.27	<0.001