LEARNER-CONDITIONED EXPERIENCE REWRITING FOR LANGUAGE MODEL AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

A successful agent trajectory demonstrates a solution, but does not identify which parts would improve the current learner. Failed trajectories can likewise contain useful actions. We introduce Learner-Conditioned Experience Rewriting (LER), a framework for converting both kinds of experience into supervision using three distinct signals: satisfaction of a checkable requirement, the learner's unaided reliability on that requirement, and the utility of a candidate update. A shared language model audits experience with training-only references and checkpoint-specific behavioral records, proposes local corrections, and learns from complete action turns admitted by explicit environment checks. Progress-conditioned replay allocates training within this verified pool. To train the rewriting policy, LER compares verified plans through equal-budget temporary updates from a common checkpoint and uses comparisons that pass relative-advantage and positive-gain filters as preference labels. The resulting objective targets measured short-horizon learning utility without differentiating through the updates. Explicit information boundaries and behavior-level measurements connect the construction, allocation, and evaluation of supervision. This formulation separates immediate correction quality from training utility, making subsequent unaided execution the objective of experience rewriting.

1 INTRODUCTION

Language model agents accumulate experience by acting in an environment, observing its responses, and attempting to satisfy a task. Such experience can support parameter learning through successful-trajectory imitation, hindsight refinement, or reinforcement learning (Zelikman et al., 2022; Zhao et al., 2025; Chen et al., 2025). Yet a trajectory's task outcome leaves an important supervision decision unresolved: which of its actions should this learner practice next? A successful solution may mostly repeat familiar behavior, while an unsuccessful attempt may already contain correct and reusable actions.

Consider a task that requires retrieving all records satisfying a condition and saving the result. An agent may query the correct endpoint, apply the requested filter, and save a file, yet omit later pages. A complete reference solves the task but also contains actions the learner already performs reliably. A targeted correction can preserve the query and filter while completing retrieval. This example exposes two further requirements. Checking that the file exists does not establish that retrieval was complete or that unrelated records were left unchanged. Even a properly checked repair does not establish that training on it will make the learner retrieve all records in a new unaided attempt.

We therefore distinguish *local validity*, *operational mastery*, and *learning utility*. Local validity means that a turn or bounded trace satisfies a declared requirement and its effect constraints. Operational mastery summarizes unaided behavior on a fixed training-side measurement population. Learning utility is the change in task performance after a specified parameter update. These quantities answer different questions: whether a target is admissible, how strongly to replay it, and which admissible teaching plan is more useful at a particular learner checkpoint.

Learner-Conditioned Experience Rewriting (LER) connects these decisions in a single training loop. A shared model uses separate execution and rewriting roles. The rewriting role receives training-only references, logged evidence, and checkpoint-specific learner records; it audits complete turns and proposes corrections to unmet requirements. Corrections are executed from restored states,

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.

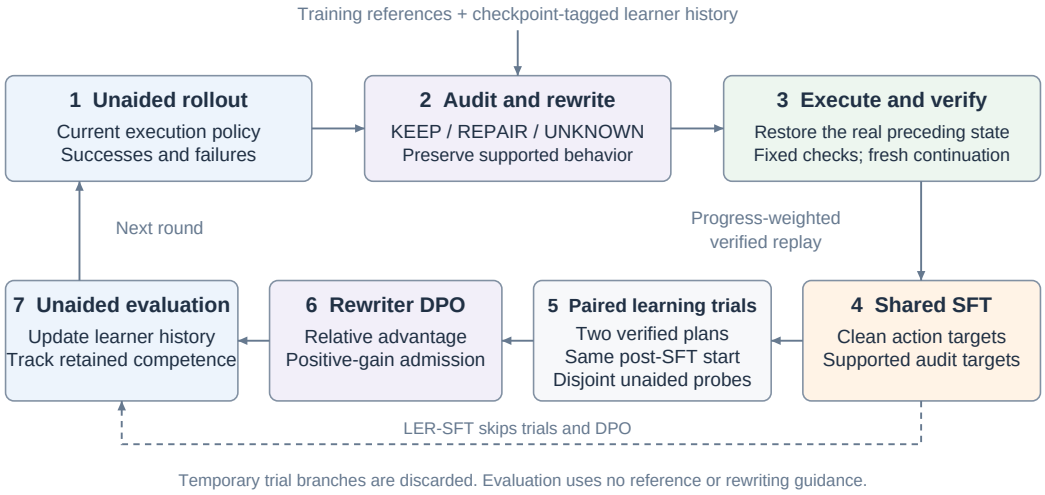

Figure 1: **Learner-conditioned experience rewriting.** Training-only references and checkpoint-specific records condition an audit and repair role. Complete execution targets pass declared requirement, effect, and provenance checks before replay and shared-model SFT. The gain stage refreshes learner records at the post-SFT checkpoint and generates new verified plans for matched trials. Admitted relative and positive-gain comparisons train plan preferences; LER-SFT skips this stage. Only task inputs and ordinary interaction history are available during unaided execution.

followed by fresh continuations, and admitted only within explicit checker coverage. The execution role is trained on task-and-history inputs with this assistance removed. Replay then changes with measured progress rather than with the source trajectory's success label alone.

The full method also learns preferences over rewriting plans. After the main supervised update, two verified candidates undergo matched temporary updates from that same checkpoint. Disjoint training-side probes measure relative utility and whether the preferred branch improves over its starting point. Supported comparisons train the rewriting role with direct preference optimization (DPO) (Rafailov et al., 2023). The finite-budget trials provide checkpoint-specific measurements of teaching utility.

Learner-dependent teaching and optimization through student performance have substantial precedents (Fan et al., 2018; Pham et al., 2021; Ren et al., 2018). Our contribution is their concrete integration with state-restorable, environment-checked agent experience: a scoped admission rule for local corrections, behavior records that separate completion from opportunity visitation, and a preference protocol over verified structured plans. The reported results evaluate the end-to-end shared LER system under unaided execution; Appendix A.4 describes protocols for decomposing direct parameter-transfer and supervision-mediated effects.

2 Related Work

Experience and hindsight in agents. ReAct interleaves reasoning and environment actions (Yao et al., 2023), while Reflexion uses verbal feedback as memory for subsequent attempts (Shinn et al., 2023). STaR bootstraps reasoning supervision from model attempts and answer-conditioned repair (Zelikman et al., 2022); SiriuS builds and refines an experience library for multi-agent learning (Zhao et al., 2025). These approaches establish the value of generated experience and corrective feedback. LER studies how to construct parameter-update targets from that experience and evaluates execution without task-specific reference text, repair instructions, or retained reflection memory.

Targeted supervision and privileged information. HinT-SD uses trajectory hindsight to localize failure-relevant actions and distills a feedback-conditioned teacher into a student operating from clean prefixes (Yeo et al., 2026). SWE-Prime filters software-agent trajectories and semantic segments using quality, contribution, local learnability, and risk (Zheng et al., 2026). Selective supervi-

sion and removal of training assistance are thus established ideas, also related to generalized distillation from privileged information (Lopez-Paz et al., 2016). LER adds explicit behavior measurements and compares environment-checked rewriting plans by post-update performance. Its learning utility is an outcome of a specified trial update, distinct from a judgment about a segment's apparent learnability.

Learning to teach and curriculum learning. Learning to Teach conditions a teaching policy on the student and optimizes it using student feedback (Fan et al., 2018). Teacher–Student Curriculum Learning allocates practice through estimated learning progress and considers forgotten tasks (Matiisen et al., 2017). Meta Pseudo Labels improves a pseudo-labeling teacher through the student's subsequent performance (Pham et al., 2021). LER builds on this teaching perspective with structured plans that produce executable corrections. Environment checks determine which parts may become action supervision, while checkpoint-specific behavioral records condition both their construction and replay.

Data utility and policy optimization. Learning to Reweight Examples uses a validation objective to guide training-example weights (Ren et al., 2018). LER likewise evaluates data through a learner update, but labels pairs of generated plans using finite-budget environment trials and fits those labels with DPO (Rafailov et al., 2023); it does not differentiate through the temporary updates. Black-box teaching rewards themselves have precedents (Fan et al., 2018). LOOP directly optimizes interactive agents with reinforcement learning (Chen et al., 2025). The LER-SFT stage instead uses checked execution and audit targets, and the full method adds preference learning in the rewriting role.

3 Learner-Conditioned Experience Rewriting

3.1 Roles, information, and behavior specifications

For task x and initial environment state s_0, let $h_t = (o_0, a_0, \ldots, o_t)$ be the history before action a_t. The execution policy samples $a_t \sim \pi_\theta(\cdot \mid x, h_t)$. An action target is a complete executable model turn, such as a tool call or a code block; its environment response is context for later turns, not a target token. An unaided rollout τ starts at s_0 and uses only the task, ordinary observations, and the policy's own actions. Its task outcome is $R(\tau)$, giving the evaluation objective

$$J(\theta) = \mathbb{E}_{x,s_0}\left[R(\mathrm{Rollout}(\pi_\theta, x, s_0))\right]. \tag{1}$$

The same weights parameterize a rewriting role $q_\theta(\cdot \mid z)$ with context $z = (x, \tau, r_x, H_\theta)$, where r_x is training-only reference information and H_θ includes learner measurements explicitly tied to checkpoint θ. This role separation restricts inputs, not parameter transfer.

A run fixes a versioned behavior registry before adaptive training. An entry

$$b = (v_b, d_b, A_b, E_b, C_b, F_b) \tag{2}$$

contains its version, semantic requirement, task-applicability rule, observable opportunity rule, requirement checker, and protected-effect contract. $A_b(x)$ determines whether the requirement applies from the task specification; E_b records where an opportunity occurs in a trace. C_b returns pass, fail, or unknown with a bounded evidence scope. F_b specifies admissible writes and state that must remain unchanged. For example, retrieving *all* matching records is a semantic requirement; using one particular pagination API is only a possible implementation. Registry entries and checkers are supplied by the experiment designer, potentially using existing environment tests. They are not assumed to follow automatically from terminal success and are counted as supervision resources.

Only registry-covered effects receive verification claims. A behavior may span multiple turns; its completion check then consumes that bounded trace, while the model is still trained on complete constituent turns with their actual prefixes. Appendix A.1 specifies the admission contract and an illustrative pagination case. The registry, splits, state-restoration procedure, and all protocol settings must be fixed and logged before the corresponding training decisions.

3.2 Grounded audits and executable corrections

The rewriting role examines both successful and failed rollouts. Its report names a behavior, affected turns, evidence locations, proposed judgment, and corrective plan. `KEEP` proposes retaining

supported behavior, REPAIR identifies an unmet requirement to address, and UNKNOWN records insufficient evidence. These are proposals: an independent evidence resolver and the registered checks determine which fields become targets. A true log citation establishes that an observation occurred; it does not by itself establish a causal explanation or a correct plan.

Corrections preserve already-satisfied requirements within the declared scope and modify unmet ones. A training reference can reveal which requirement is missing, but a retained execution target must be supported by the task and its own clean preceding history. Its task-specific constants, object selections, and dependencies require provenance in that prefix; future observations and hidden reference answers cannot supply missing inputs. When a necessary fact is unavailable, the rewrite must acquire it through ordinary actions. Unresolved provenance excludes that target.

Each candidate runs from the state immediately preceding its edited turn, restored by snapshot or exact prefix replay. A restoration check compares the relevant state and environment version. All observations and subsequent actions are then collected afresh; a corrected prefix is never attached to an obsolete suffix. A candidate enters the action pool only if the applicable requirement checks and protected-effect checks pass over their declared scope, provenance is resolved, and no covered effect remains unknown. Final task success is recorded separately. Thus a locally admitted action may come from a globally failed task, without being represented as a complete successful solution.

For audit SFT, validated fields are serialized into a canonical report. Unsupported fields are replaced by UNKNOWN or removed *before* teacher forcing, preventing unsupported text from becoming a conditioning prefix for later target tokens. Factual fields are supported by resolved evidence and checker outputs; a proposed repair is supervised as a plan only after its execution is admitted. Unchecked explanations are not promoted to audit truth. This produces separate execution and audit datasets with explicit evidence provenance.

3.3 Comparable behavior records and shared SFT

Operational mastery. To keep the measurement population stable, a run fixes a stratified training-side panel $\mathcal{M}$ of initial tasks and weights $\rho_{b\ell}$ for each applicable stratum ℓ, with $\sum_\ell \rho_{b\ell} = 1$. Every checkpoint is evaluated from the initial states without rewriting assistance. For each of V registered rollout repetitions, let $Y_{\theta b}^{(v)}(x) = 1$ if the complete rollout satisfies requirement b, and zero otherwise; write $\overline{Y}_{\theta b}(x) = V^{-1} \sum_{v=1}^{V} Y_{\theta b}^{(v)}(x)$. Unreached requirements remain in the denominator; unknown checks are reported separately and conservatively count as zero for scheduling. A valid alternative route can still satisfy C_b. The panel's certified-completion estimate is

$$\widehat{p}_\theta(b) = \sum_\ell \rho_{b\ell} \frac{\sum_{x \in \mathcal{M}_{b\ell}} \overline{Y}_{\theta b}(x)}{|\mathcal{M}_{b\ell}|}. \tag{3}$$

Opportunity reach rates and completion conditional on reaching an opportunity are additional diagnostics. They do not replace the fixed-denominator measure. Accordingly, "mastered" is a panel-relative scheduling state for end-to-end requirement satisfaction, not a claim about context-free skill or unseen-task transfer.

The history records $\widehat{p}_\theta(b)$, its uncertainty, opportunity composition, and actual training exposure. At prespecified measurement checkpoints, a behavior is *unresolved* when coverage is insufficient; *mastered* when its lower uncertainty bound exceeds a fixed threshold over the required window; *deferred* when sufficient recent exposure accompanies persistently limited progress; and otherwise *acquiring*. Positive maintenance weights preserve replay for mastered and deferred behaviors. Deferral has a fixed expiry, after which practice resumes even without spontaneous improvement. Appendix A.2 defines the transition order and measurements.

Complete-turn replay. An admitted example i consists of clean input $c_i = (x_i, h_i)$, complete action a_i, and a nonempty behavior-label set B_i. Let $w_\theta(b) > 0$ be the registered state weight, with acquiring behaviors emphasized over mastered and deferred ones.

$$\omega_\theta(i) = \frac{1}{|B_i|} \sum_{b \in B_i} w_\theta(b), \qquad P_\theta(i) = \frac{\omega_\theta(i)}{\sum_{i'} \omega_\theta(i')}. \tag{4}$$

This gives each turn one sampling priority; it neither duplicates a multi-behavior turn nor guarantees equal total exposure across behaviors. Exposure is measured from actual sampled turns and tokens for every associated label.

Only execution-target tokens contribute to

$$\mathcal{L}_{\text{exec}}(\theta) = -\mathbb{E}_{i\sim P_k}\left[\frac{1}{|a_i|}\sum_{j=1}^{|a_i|}\log \pi_\theta(a_{i,j} \mid c_i, a_{i,<j})\right]. \tag{5}$$

Here P_k is frozen for the round from its most recent checkpoint-tagged history. The clean execution input excludes references, audit reports, repair instructions, and hidden environment state. The audit loss averages log loss over supported tokens of the sanitized reports under the rewriting context. The shared update minimizes

$$\mathcal{L}_{\text{SFT}} = \mathcal{L}_{\text{exec}} + \lambda_{\text{audit}}\mathcal{L}_{\text{audit}}, \qquad \phi_k = \text{SFT}(\theta_k; \mathcal{D}_k). \tag{6}$$

This stage defines LER-SFT. Its checkpoint ϕ_k is the common anchor for the optional gain stage.

3.4 Preferences from controlled learning trials

The full method refreshes learner measurements at ϕ_k before constructing gain contexts $z = (x, \tau, r_x, H_{\phi_k})$. Two plans $y_A, y_B \sim q_{\phi_k}(\cdot \mid z)$ specify audits, repairs, and selected supervision. Executing and admitting them produces D_A, D_B. Both must pass the same contract before utility is compared. The canonical plan serialization applies the same unsupported-field removal as audit SFT. The plans and their candidate data are frozen before probe outcomes are observed.

For each registered trial seed r, both branches clone ϕ_k and its optimizer state. They use the same update operator U, background set B, candidate/background token allocation, and total supervised-token budget K:

$$\theta_{jr} = U_K(\phi_k; B, D_j, r), \qquad j \in \{A, B\}. \tag{7}$$

Differences in candidate data are the treatment; the batch construction and matching rules are fixed. Branches are temporary and are discarded after scoring. On a fresh training-side probe block Q, disjoint from candidate and background task families, let $\widehat{J}_{Q,r}$ average task-level rollout repeats with fixed stratum weights and evaluation randomness paired within trial replicate r. The utility estimates are

$$\widehat{g}_j = \frac{1}{m}\sum_{r=1}^{m}\left[\widehat{J}_{Q,r}(\theta_{jr}) - \widehat{J}_{Q,r}(\phi_k)\right], \qquad \widehat{\delta} = \widehat{g}_A - \widehat{g}_B. \tag{8}$$

The anchor baseline cancels from $\widehat{\delta}$: a relative winner need not have positive utility. We therefore retain $A \succ B$ only when

$$L(\delta) > \epsilon \quad \text{and} \quad L(g_A) > 0, \tag{9}$$

and symmetrically retain $B \succ A$ using $L(-\delta) > \epsilon$ and $L(g_B) > 0$. L denotes the prespecified lower uncertainty bound and ϵ the minimum useful relative margin. Comparisons receive preference labels only when both admission conditions hold. Paired evaluation randomness reduces evaluation noise; independent trial repetitions are still needed to assess update variability. Appendix A.3 defines the replication, simultaneous-bound, and probe-use protocol. This empirical admission rule identifies preferences supported under the registered checkpoint, probe distribution, and finite update budget.

For admitted pairs $\mathcal{P}_k$, freeze $q_{\text{ref}} = q_{\phi_k}$, initialize the main model at ϕ_k, and optimize plan-token DPO:

$$\mathcal{L}_{\text{gain}}(\theta) = -\mathbb{E}_{(z,y^+,y^-)\sim\mathcal{P}_k}\log\sigma\left(\beta\left[\log\frac{q_\theta(y^+ \mid z)}{q_{\text{ref}}(y^+ \mid z)} - \log\frac{q_\theta(y^- \mid z)}{q_{\text{ref}}(y^- \mid z)}\right]\right). \tag{10}$$

The main SFT, candidate generation, trial anchors, and DPO reference thus have an explicit order. If no pairs qualify, the round returns ϕ_k. Otherwise it returns the DPO-updated shared checkpoint. Each preference is defined by its recorded anchor and finite trial budget. As DPO also changes the execution policy's weights, a subsequent task-score change measures both direct parameter transfer and any later effect of altered supervision. Appendix A.4 describes controls for decomposing these effects.

Algorithm 1 One round of learner-conditioned experience rewriting

Require: Shared checkpoint θ_k, checkpoint-tagged history, fixed registry and protocol
Collect unaided trajectories; measure behavior on the fixed training-side panel
Audit and propose corrections using training references and learner records
Restore states, execute corrections, and collect fresh continuations
Admit complete action targets and sanitized audit targets with recorded check evidence
Set replay priorities; update the main model by SFT to obtain ϕ_k
Initialize $\theta_{k+1} \leftarrow \phi_k$
if learning-gain preference training is enabled **then**
    Refresh behavior measurements at ϕ_k; generate pairs of verified plans
    Clone ϕ_k and its optimizer state for matched, repeated trial updates
    Score branches and anchor on reserved unaided probes; apply Eq. 9
    Discard branches; freeze $q_{\text{ref}} = q_{\phi_k}$
    if admitted pairs exist **then**
        Update θ_{k+1} by plan-token DPO initialized at ϕ_k
    end if
end if
Evaluate θ_{k+1} unaided; append checkpoint-tagged outcomes and exposure
return $\theta_{k+1}, H_{\theta_{k+1}}$ and the round's resource ledger

4 EXPERIMENTS

Our evaluation compares unaided task completion at two backbone scales on three benchmarks, then examines learner-conditioned supervision, audit updating, learning-gain preferences, and replay through design variants.

4.1 BENCHMARKS AND EVALUATION PROTOCOL

Tasks and metrics. AppWorld reports task-goal completion (TGC) separately on its Normal and Challenge splits (Trivedi et al., 2024). BFCL-v3 evaluates multi-turn function calling (Patil et al., 2025); our MT metric macro-averages accuracy across Base, Missing Functions, Missing Parameters, and Long Context, requiring state and response checks to pass at every turn. It is distinct from BFCL Composite and overall scores. ALFWorld reports unseen-task success rate for text-based embodied instruction following (Shridhar et al., 2021). The four columns represent three benchmarks; all values are percentages, with no cross-column average.

Backbones and evaluation. We use Qwen3-4B and Qwen3-8B, sharing the initial checkpoint within each scale. Evaluation starts from the original task state with only the task and ordinary observations. We report Avg@8 scores at a decoding temperature of 0.6; methods share the harness, action budget, and decoding settings. References, audit reports, repair guidance, and task-specific reflection memories are excluded. Training probes used for preference labels are disjoint from final test tasks.

Comparison methods. *Initial policy* is the starting checkpoint before additional training. *Iterative success-only SFT* alternates policy rollout collection with supervised fine-tuning on the successful rollouts. *GRPO* optimizes terminal rewards (Shao et al., 2024); *SFT → GRPO* applies supervised fine-tuning followed by GRPO. We adapt *HinT-SD-Multi* and *SWE-Prime* to the evaluation environments, marking these implementations with † in Table 1. HinT-SD-Multi retains multi-turn failure localization and feedback-conditioned targeted reverse-KL supervision (Yeo et al., 2026). SWE-Prime retains two-level successful-trajectory filtering and segment-masked training with surrounding context (Zheng et al., 2026); its contribution, local-learnability, and risk criteria are mapped to each environment's task requirements. LER combines verified execution/audit SFT with rewriting-plan DPO, without execution-side reinforcement learning.

Fair budgets. Methods share permitted training information and environments. A common total budget covers collection, auditing, repair, verification, temporary learning trials, probes, and final

Table 1: **Main results on three agent benchmarks (Avg@8).** AppWorld reports task-goal completion (TGC) on Normal and Challenge; BFCL-v3 reports multi-turn (MT) accuracy; ALFWorld reports unseen-task success rate (SR). All scores are percentages under unaided evaluation at temperature 0.6; higher is better. Bold indicates the best result per metric within each backbone. † denotes an environment-adapted implementation, as described in the comparison-methods paragraph.

	AppWorld (TGC)		BFCL-v3	ALFWorld
Method	Normal ↑	Challenge ↑	MT acc. ↑	Unseen SR ↑
Backbone: Qwen3-4B				
Initial policy	10.38	3.74	26.35	52.13
Iterative success-only SFT	20.36	8.62	34.34	67.43
GRPO	22.23	9.58	36.85	70.19
SFT → GRPO	25.73	12.14	39.98	74.73
HinT-SD-Multi†	27.14	13.80	43.51	75.65
SWE-Prime†	26.90	12.87	41.73	76.38
LER (ours)	**29.57**	**15.55**	**45.65**	**78.18**
Backbone: Qwen3-8B				
Initial policy	18.90	6.65	34.53	61.41
Iterative success-only SFT	31.71	15.15	43.32	76.24
GRPO	33.89	16.77	45.42	79.21
SFT → GRPO	38.62	20.49	50.23	82.44
HinT-SD-Multi†	40.66	22.25	52.92	83.18
SWE-Prime†	39.62	21.06	51.43	83.82
LER (ours)	**42.76**	**24.06**	**54.59**	**86.13**

updates; SFT → GRPO shares one cap across both stages. Tokens, environment calls, and GPU time are recorded separately.

4.2 Main results

As shown in Table 1, LER leads all four metrics at both scales. HinT-SD-Multi is the strongest baseline on AppWorld and BFCL-v3, while SWE-Prime is strongest on ALFWorld. Relative to these column-wise comparators, LER has margins of 2.43, 1.75, 2.14, and 1.80 percentage points for Qwen3-4B, and 2.10, 1.81, 1.67, and 2.31 points for Qwen3-8B, in table order.

SFT → GRPO exceeds either objective alone on all four metrics at both scales. Relative to this combination, LER adds 3.84, 3.41, 5.67, and 3.45 points at 4B, and 4.14, 3.57, 4.36, and 3.69 points at 8B. These comparisons target supervision construction beyond success-based imitation and terminal-reward optimization. The strongest adapted baseline in each column provides a closer comparison, with margins of 1.67–2.43 points across the two scales.

4.3 Component analysis

Table 2 compares design variants on Qwen3-8B, with full LER as A0. Each row removes or replaces the indicated mechanism; the descriptions below specify which operations are retained. All variants use unaided Avg@8 evaluation at temperature 0.6. Differences below are percentage-point reductions from A0 in Table 2 and characterize the corresponding design change, including its effects on the training data and updates.

Learner-history input and repair generation (A1–A2). A1 removes checkpoint-specific learner-history input from the rewriter while retaining source trajectories, verification, and adaptive replay. Scores decrease by 2.38–2.81 points, supporting the use of learner information in rewriting; this variant does not remove learner conditioning from replay. A2 disables repair generation and selects only verified original segments, while retaining preference comparisons between alternative selections. Scores decrease by 3.07–3.65 points, consistent with a benefit from generated repairs beyond selecting existing experience; the largest reduction is on BFCL-v3.

Table 2: **Component analysis of LER on Qwen3-8B (Avg@8).** Variants are defined relative to full LER (A0): "w/o" removes an input, operation, or data source; "w/" replaces the corresponding strategy. Metrics, units, and evaluation temperature follow Table 1. A4 is LER-SFT, which omits both learning trials and preference DPO. A7 changes action-target filtering while retaining diagnostic checks and audit-evidence rules. Bold indicates the best result per metric.

		AppWorld (TGC)		BFCL-v3	ALFWorld
ID	Variant	Normal ↑	Challenge ↑	MT acc. ↑	Unseen SR ↑
A0	**LER (full)**	**42.54**	**24.23**	**54.65**	**85.95**
A1	w/o learner history in rewriting	39.76	21.42	51.86	83.57
A2	w/o repair generation	39.18	20.87	51.00	82.88
A3	w/ frozen initial auditor	41.08	22.59	53.15	84.76
A4	LER-SFT (w/o gain stage)	40.75	22.47	52.75	84.43
A5	w/ uniform replay	41.26	22.87	53.46	84.76
A6	w/o failed-source action targets	40.55	22.03	52.53	84.24
A7	w/ terminal-success filtering	39.57	20.52	51.61	82.37

Audit updates and the gain stage (A3–A4). A3 uses a frozen copy of the initial checkpoint for audit-report generation. Execution and repair generation use the current model, and both SFT objectives and preference training remain active. Scores decrease by 1.19–1.64 points. A4 is LER-SFT (Section 3.3): it omits the gain stage, including temporary learning trials and preference DPO, while retaining verified execution/audit SFT. Its reductions of 1.52–1.90 points compare the complete stage with its omission, rather than isolating preference labels or DPO alone.

Replay distribution and failed-source experience (A5–A6). A5 replaces adaptive replay with uniform sampling over eligible targets, reducing scores by 1.19–1.36 points. A6 removes all action targets originating from failed source trajectories, including their locally valid actions and subsequently successful repairs. Source success is determined before repair. Scores decrease by 1.71–2.20 points. These variants examine replay allocation and access to failed-source experience, respectively.

Action-target filtering (A7). A7 replaces behavior-level admission with terminal-success filtering for action supervision, while preserving actual-state execution and information safeguards. Behavior checkers still provide diagnostics, and audit targets retain their evidence rules. This rule excludes locally valid turns from trajectories that do not achieve terminal success; a repaired trajectory that succeeds may still supply targets. Scores decrease by 2.97, 3.71, 3.04, and 3.58 points on AppWorld Normal, AppWorld Challenge, BFCL-v3, and ALFWorld, respectively. The comparison evaluates an alternative supervision-selection rule, including its effect on the eligible data pool, rather than removing verification throughout the system.

The behavioral protocol pairs repair validity with subsequent unaided completion, fixed-checker reliability, exposure, and replay allocation. Preference acceptance and paired probe uncertainty characterize the training signal supplied to the rewriter.

5 Conclusion

Learner-Conditioned Experience Rewriting treats agent experience as supervision whose usefulness depends on the learner. It separates scoped validity, measured requirement completion, and post-update utility: environment checks determine admission, checkpoint-specific records guide rewriting and replay, and controlled trials supply preferences among verified plans. The shared model learns executable actions from clean histories and learns rewriting preferences from their measured effects on a temporary learner. This connects local correction to an explicit learning objective and makes successful and failed experience available within the same supervision pipeline. The framework provides a concrete basis for studying when checked corrections become reliable unaided behavior, with verification coverage, learning utility, and total resource use measured as distinct quantities.

6 LIMITATIONS

LER depends on restorable environments and a registry of requirements with tractable checks. The registry encodes designer knowledge, and a passing check supports only its stated scope. Sparse coverage, incorrect effect contracts, or unresolved information dependencies can either admit poor targets or exclude useful experience. Provenance verification covers dependencies represented by the registered evidence model.

Operational mastery is defined relative to the fixed behavioral panel and its task composition. It measures end-to-end requirement satisfaction, which can change through upstream behavior, and repeated use can adapt the learner to the panel. Rare behaviors may remain unresolved. Maintenance replay and bounded deferral preserve opportunities for further practice; retention and transfer beyond this population are separate evaluation targets.

Learning trials add optimization, rollout, and verification cost. Small utility differences may remain unresolved within an affordable budget; abstention can then leave little preference data. Learning-gain preferences are conditioned on the checkpoint, update budget, probe distribution, and uncertainty procedure. Longer-horizon stability is a separate evaluation target. Shared parameters motivate separate attribution analyses for direct parameter-transfer and supervision-mediated effects.

AI USE STATEMENT

Generative AI tools assisted with manuscript drafting, LaTeX preparation, literature lookup, and refinement of the experimental protocol. The authors are responsible for the manuscript's technical content, citations, and experimental claims.

REFERENCES

Kevin Chen, Marco Cusumano-Towner, Brody Huval, Aleksei Petrenko, Jackson Hamburger, Vladlen Koltun, and Philipp Krähenbühl. Reinforcement learning for long-horizon interactive LLM agents. *arXiv preprint arXiv:2502.01600*, 2025. URL https://arxiv.org/abs/2502.01600.

Yang Fan, Fei Tian, Tao Qin, Xiang-Yang Li, and Tie-Yan Liu. Learning to teach. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HJewuJWCZ.

David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. Unifying distillation and privileged information. In *International Conference on Learning Representations*, 2016. URL https://arxiv.org/abs/1511.03643.

Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher-student curriculum learning. *arXiv preprint arXiv:1707.00183*, 2017. URL https://arxiv.org/abs/1707.00183.

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 *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392, 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

Hieu Pham, Zihang Dai, Qizhe Xie, and Quoc V. Le. Meta pseudo labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11557–11568, 2021. URL https://openaccess.thecvf.com/content/CVPR2021/html/Pham_Meta_Pseudo_Labels_CVPR_2021_paper.html.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023. URL https://arxiv.org/abs/2305.18290.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4334–4343, 2018. URL https://proceedings.mlr.press/v80/ren18a.html.

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. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*, 2023. URL https://arxiv.org/abs/2303.11366.

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

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024. URL https://appworld.dev/appworld/.

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

Woongyeong Yeo, Yumin Choi, Taekyung Ki, and Sung Ju Hwang. HinT-SD: Targeted hindsight self-distillation for long-horizon agents. *arXiv preprint arXiv:2605.17873*, 2026. URL https://arxiv.org/abs/2605.17873.

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.

Wanjia Zhao, Mert Yuksekgonul, Shirley Wu, and James Zou. SiriuS: Self-improving multi-agent systems via bootstrapped reasoning. *arXiv preprint arXiv:2502.04780*, 2025. URL https://arxiv.org/abs/2502.04780.

Dewu Zheng, Ruizhe Ye, Yanlin Wang, Yang Ye, Hongyu Zhang, Ensheng Shi, Xilin Liu, Yuchi Ma, Jianxing Yu, and Zibin Zheng. SWE-Prime: Fewer trajectories, better performance. *arXiv preprint arXiv:2608.27449*, 2026. URL https://arxiv.org/abs/2608.27449.

A Technical Protocol

This appendix specifies behavior contracts, measurement schedules, learning trials, and resource accounting for LER. A run manifest fixes the symbolic settings before the decisions they govern and connects those settings, code versions, and data identifiers to each reported result.

A.1 Behavior contracts and admission

Registry construction. For each b in Eq. 2, the registry records the author or generation-and-review process; task families for which A_b applies; a machine-evaluable opportunity predicate E_b; checker inputs and outputs; covered turns or trace interval; and the effect contract F_b. Applicability follows the task's semantic requirement, not a preferred solution path. Path-specific operations may be recorded as diagnostics but cannot define end-to-end mastery when another legal solution avoids them. Changing any of these definitions creates a new version and a separate measurement series. Proposed new behaviors cannot silently change the denominator of an existing series.

What is checked. A contract consists of a postcondition on the declared requirement, invariants that hold throughout its relevant trace, and a frame condition on protected state. The environment adapter must expose enough trace and state information to evaluate that contract. Admission requires agreement of all applicable checks within this scope; an exception, missing state, unsupported action effect, or indeterminate result yields `UNKNOWN` and prevents admission. An environment with only a terminal evaluator can support terminal-verified data, but does not thereby support the local-admission claim used by LER.

For a turn with several effects, B_i contains the registered requirements whose contracts jointly cover those effects. Each associated contract must pass; a passing check for one effect cannot authorize unrelated operations in the same turn. If coverage is incomplete, the rewriter may propose simpler complete turns, which must be executed and checked again. For a multi-turn behavior, the enclosing bounded trace must pass and each admitted constituent turn must satisfy its effect contract. Span-level success supports the trace's outcome, not a claim that every constituent action was necessary or caused that outcome. Verification is scoped to the state projection and effects covered by the registered contract.

Illustrative retrieval contract. Suppose the requirement is to save exactly the records matching a user predicate. A supported adapter would need a training-side oracle for the expected set, a comparison between that set and the saved contents, a check that the user-specified location is used, and a state-difference check for protected objects. Opportunity evidence could include an observed continuation token. Failure to visit a later page is not automatically an error if a different legal call returns the complete set; the semantic completion check decides. A file-existence check alone does not cover completeness, filtering, or protected state. This illustrative contract specifies the oracle interfaces that an environment adapter would need to support the retrieval requirement.

Audit targets and clean inputs. An audit record has fields for registry version, source trajectory and turn identifiers, evidence locations, observed checker outcomes, judgment, and proposed edit. The evidence resolver checks the cited payloads; checker outcomes supply effect labels. Unsupported explanations are replaced by `UNKNOWN` before serialization. A field-level mask then restricts audit loss to admitted content. Reference-conditioned input is permitted for this role, so the loss does not imply that audits themselves are unaided. The same canonical representation is used for plan-token DPO.

An execution example stores the restored-state identifier, executable action, actual clean prefix, associated contracts, and provenance records. Provenance must cover object identifiers, literal values, selections, and prerequisite observations, not just remove reference strings. Prefix replay or a clean-input reconstruction can test dependencies represented by the available provenance evidence. Unresolved dependencies trigger abstention. A repair plan may request a missing observation; the resulting ordinary observation can support later action targets. Generated continuations remain separate observations until their own targets pass admission.

A.2 Measurement, state transitions, and replay

Fixed population and uncertainty. The manifest fixes the panel $\mathcal{M}$, task-family strata, applicability lists, weights $\rho_{b\ell}$, rollout budget, decoding settings, evaluation-seed schedule, and measurement checkpoints. At every checkpoint all applicable panel tasks enter Eq. 3, including failures before a relevant opportunity. The log distinguishes pass, observed fail, requirement not reached, and checker unknown; valid alternative routes are evaluated by the same semantic requirement. It separately reports opportunity counts, reach rates, conditional completion, and the distribution of reached contexts. These quantities reveal routing changes instead of interpreting every conditional-rate change as learning.

The run fixes a task-family-clustered uncertainty procedure and confidence level before scheduling begins. We specify a paired stratified cluster bootstrap: resample task families within each registered stratum, retain all tasks and rollout repeats in each sampled family, and recompute the weighted estimate. For checkpoint differences, use the same resampled families at both checkpoints. Repeated decodes of one task are not independent tasks. Checkpoint times and any multiplicity adjustment are fixed in advance. Such intervals quantify variability under the specified resampling units, with the fixed training panel serving as the measurement population for adaptive scheduling.

Operational states. Let t index registered measurement events, separately from the outer training round k; each event records its checkpoint, including measurements of θ_k and ϕ_k. Write $\widehat{p}_t(b)$ for the estimate at that event and $[L_t(b), U_t(b)]$ for its uncertainty interval. Let W count measurement events in the progress window, $n_{\min}$ denote the minimum supported task-family coverage, $p_\star$ the mastery threshold, and $E_\star$ a recent exposure threshold. Define $\Delta_t(b) = \widehat{p}_t(b) - \widehat{p}_{t-W}(b)$ with paired interval $[L_t^\Delta(b), U_t^\Delta(b)]$. The transition order is:

1. Set *unresolved* if required panel strata are missing or supported task-family coverage is below $n_{\min}$. Unknown outcomes remain visible and never count as successful opportunities.
2. Otherwise set *mastered* when lower bounds exceed $p_\star$ at every checkpoint in the registered persistence window, provided that window is available.
3. For other behaviors, a previously mastered behavior whose mastery criterion has failed returns to *acquiring*. An expired deferral also forces acquiring for at least $A_{\min}$ outer rounds. Deferral and cooldown counters advance only at outer-round boundaries; an intermediate ϕ_k measurement does not advance them.
4. Otherwise set *deferred* when no acquiring cooldown is active, the progress window is available, recent exposure exceeds $E_\star$, and $U_t^\Delta(b) \leq \eta$, where η is the prespecified small-progress tolerance. Each deferral lasts at most $D_{\max}$ outer rounds and is not extended by repeated low-progress measurements. Deferral is a temporary scheduling state.
5. Set the remaining eligible behaviors to *acquiring*.

The manifest fixes all windows, thresholds, and four positive state weights before the run. Values are selected from a finite declared configuration set on training/development tasks under a fixed tuning budget, using development task performance subject to the resource cap; ties use lower cost and then a predetermined configuration order. Test outcomes cannot choose these settings. This finite-state scheduler is one concrete instantiation of learner-conditioned replay.

Exposure accounting. Sampling follows Eq. 4 with a fixed budget and batch rule. For every sampled example, record its complete-turn target-token count, number of presentations, and every label in B_i. Scheduling exposure counts execution-target presentations on the main model's parameter path; discarded trial branches do not increase it. Audit and DPO tokens are recorded separately. Behavioral exposure totals can overlap because the same turn trains several behaviors; summing them is not a global token ledger. The actual global token count counts that turn only once. A behavior with many examples can receive more aggregate exposure than another with the same weight, so both realized exposure and the behavior composition of the pool must accompany reported replay weights.

A.3 Trial replication and preference admission

Data partitions and time order. The manifest partitions training data by the strongest available dependence unit, such as task family or shared scenario, into rollout/background data, the behavioral panel, and reserved utility-probe blocks. Probe blocks are excluded from candidate and background updates; development data choose configurations, and final test data are used only for the final evaluation. Candidate pairs, their verifier outputs, and their target datasets are frozen before their assigned block is scored. Blocks are assigned in advance, used for their registered comparison batch once, and retired after preference labels are generated. Scores cannot trigger additional candidate search on that same block. A finite block supply and trial budget determine when preference collection stops.

Each gain context carries the ϕ_k identifier and fresh H_{ϕ_k} measurements. An original source trajectory may be older, but its source checkpoint is recorded and is not substituted for the current learner's state. Trials clone the same post-SFT parameter and optimizer states; the DPO reference and initial DPO parameters are also ϕ_k. The candidate plans are not inserted into the main SFT update before their own comparison. They may enter later replay only through the same admission rule as other experience. Trial branches never become the main checkpoint.

Matched update operator. Fix the number of trial steps, target-token budget K, candidate-token fraction, background batches, optimizer configuration, and seed set before measurement. A deter-

ministic packing and recycling rule allocates the same number of candidate and background target tokens to each branch without splitting a complete-turn target; unused budget caused by packing is matched across branches and reported. Empty datasets or pairs that cannot meet the registered matching tolerance abstain. Execution and admitted audit targets, if both are included, use the same component allocation in each branch. The final learner update need not equal this short operator, which is why utility labels are specific to U_K.

Uncertainty and abstention. Use $m \geq 2$ independently seeded matched trial pairs and a fixed number of paired rollout repeats for each probe task; the chosen m must support the registered update-variability estimate. Within a pair, clone optimizer state and match background order and evaluation randomness. Candidate execution randomness is either fixed when D_A, D_B are frozen or explicitly replicated as an additional recorded level. The anchor, A, and B use identical evaluation settings and task instances.

The specified uncertainty procedure is a paired two-factor bootstrap: independently resample trial-seed pairs and probe task-family clusters, retain the full crossed set selected by those draws, and average paired task outcomes with the registered stratum weights. Evaluation repeats remain grouped within their tasks. For a batch of N predeclared comparisons and nominal error level α, use the bootstrap $\alpha/(4N)$ quantile as the lower bound for each of g_A, g_B, δ, and $-\delta$. This Bonferroni allocation accounts for selecting a direction and a positive-gain branch within that batch. The bootstrap size, N, and α are fixed before scoring. If independent seeds, task families, or bootstrap resolution are insufficient for the registered bounds, the comparison is unresolved. These empirical bootstrap bounds quantify uncertainty within each predeclared comparison batch under the registered trial and resampling protocol.

Equation 9 requires both a supported relative difference and supported positive anchor-relative gain. Rejected pairs receive no DPO label; their checks and scores remain in the ledger. Report the proposed, verification-passing, compared, and admitted pair counts; both-negative branch frequency; gain and margin distributions; and preference reversals across independent trial repetitions. Predictive validity can be evaluated by comparing short-trial preferences with independently measured downstream updates.

A.4 Attribution and resource accounting

What a system score identifies. Because q_θ and π_θ share weights, the following controls provide a protocol for decomposing direct parameter-transfer and supervision-mediated effects. To measure direct parameter transfer, evaluate ϕ_k and the post-DPO checkpoint on matched unaided tasks before further SFT. To measure a supervision-mediated difference, freeze old and new rewriters, generate equal-budget verified datasets from the same source tasks, and train identical copies of a fixed learner on those datasets. A gain in the latter comparison supports better teaching under that controlled learner and budget. Complementary controls include utility-based selection without DPO, an equal-cost extra-SFT branch, and candidate-matched preference controls.

Resource vector. For each stage s—collection, audit, repair, restoration/verification, temporary updates, probes, main SFT, and DPO—record

$$\mathbf{c}_s = (T_s^{\text{in}}, T_s^{\text{out}}, T_s^{\text{target}}, N_s^{\text{env}}, N_s^{\text{check}}, G_s, H_s), \qquad \mathbf{c}_{\text{total}} = \sum_s \mathbf{c}_s. \tag{11}$$

These components are model input/output tokens, supervised target tokens, environment calls, checker invocations, accelerator time with hardware identity, and human specification/review time. Include rejected candidates, failed restoration, discarded trial branches, and probe evaluation. Report wall-clock time and peak memory separately because concurrent stage times do not add in the same way. Behavior-level overlapping exposure counts are not substituted for T_s^{target}.

The manifest declares componentwise caps, stopping rules, and whether saved resources in an ablation are reallocated. A same-data or same-token comparison answers a different question from a same-total-compute comparison; neither should be described as satisfying all of them. Information matching records the reference fields and checker outputs actually supplied to each method, not only the information it is allowed to access. Cross-domain adaptations require their own prompts,

mapping rules, and tuning budgets. Alongside performance, report cost per admitted target and preference and performance as a function of the chosen budget.

Run manifest. A reproducible instantiation identifies model and tokenizer revisions; environment, registry, prompt, and code versions; split and family identifiers; rollout/action limits; restoration checks; target masks and packing rules; all scheduler and trial settings; SFT/DPO optimizer and parameter-update configuration; seeds; checkpoint-selection rules; and result-to-run mappings. Checker coverage and errors require a separately reviewed sample that records unsupported admissions, false rejections, and the scope left unknown. In this protocol, the run manifest is the record for an instantiation's concrete numerical settings, implementation versions, and diagnostic outcomes.