WHEN DOES MEMORY BECOME CAPABILITY? AGENT SELF-IMPROVEMENT THROUGH MEMORY EVOLUTION AND CONSOLIDATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic reinforcement learning (RL) trains LLM agents from multi-turn environment interaction, but each episode reaches the policy only through a scalar outcome reward. Procedural memory offers a complementary channel: agents store natural-language guidance from past interactions and reuse it on future tasks. However, a memory that helps its source episode may not help on other tasks, and a memory that helps when retrieved may not improve the agent when used for training. We introduce MECA (Memory Evolution and Consolidation for Agents), which brings memory management into agentic RL: a single policy learns to solve tasks and to maintain its procedural memory by adding, revising, and removing memories. MECA rewards each proposed memory edit by its transfer, the change in success it causes on tasks it was not written from, and optimizes acting and editing jointly in one RL update. The same signal decides whether a memory stays in external memory or is consolidated into the parameters through self-distillation; the updated agent then generates new experience and edits. Memory management thus becomes a learned component of agent self-improvement. Across ALFWorld, WebShop, and Search-QA with three backbones, MECA outperforms controlled agentic RL and memory-based baselines, including GiGPO trained on as many environment trajectories, learns to write increasingly transferable memories, and retains most of its gains when all memory is removed at inference.

1 INTRODUCTION

A self-improving agent should get better at future tasks because of the tasks it has already done. Whether it browses the web, operates software, or explores an open world, interaction reveals what instructions leave unstated: which actions and strategies work, and how to recover from mistakes (Yao et al., 2023; Wang et al., 2024a). Such lessons can live in memory or in parameters: *procedural memory* keeps them as natural-language guidance retrieved on later tasks (Shinn et al., 2023; Zhao et al., 2024; Wang et al., 2024b), and recent methods also train it into the policy (Shi et al., 2026; Zhang et al., 2026c; Tu et al., 2026; Wu et al., 2026; Lin et al., 2026).

Such an agent repeatedly faces two decisions: *which memories to write and keep*, and *which of them to learn into its parameters*. Both require judging a memory's usefulness: SkillCAT measures *source benefit* on source-task replays (Chen et al., 2026b), while SAPO evaluates skill utility on queries that also supply the induction evidence (Zhang et al., 2026c). Yet source benefit can reward episode-specific notes: in a household task, "the mug is in cabinet 3" explains the episode that produced it but is useless elsewhere, whereas "search cabinets before countertops when looking for a mug" helps across many tasks. Moreover, a memory can raise success while the agent reads it yet fail to improve the policy once trained into its parameters; internalization is known to be hard (Chen et al., 2026a). What is missing is a criterion for which memories generalize and which are worth training on.

To find one, we separate two ways a memory pays off (Figure 1): *read value*, the change in success when a fixed policy reads it, and *learning value*, the change in memory-free success from training on it, versus equal ordinary training. Read value on the memory's source task is its source benefit; on tasks it was not written from, its *transfer*. Section 2 shows that source benefit poorly predicts transfer

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.

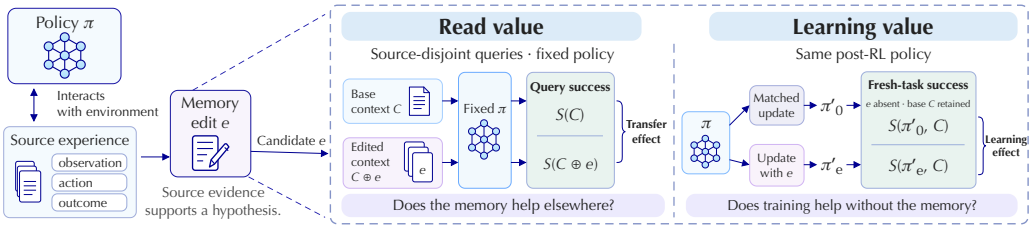


Figure 1: **Two ways a memory edit e can pay off.** *Read value* compares a fixed policy with context C and $C \oplus e$ on other tasks; *learning value* compares training on e with a matched update, both evaluated with e removed. S is the success rate (V in the text).

(39% of edits that help their source hurt other tasks) and that transfer, though only loosely tracking learning value, selects memories to consolidate better than random or source-based selection. This suggests one criterion for both decisions: *a memory becomes capability when it transfers*.

Building on these findings, we introduce MECA (Memory Evolution and Consolidation for Agents), in which a single model learns to act, edit its procedural memory, and consolidate selected memories into its parameters. Each training batch is split into source and query tasks. From source trajectories, the agent proposes an edit that adds, revises, or deletes a memory; the edit is inserted into the contexts of the query tasks, and the resulting change in success, its transfer, is the edit's reward. The same estimate routes the edit: edits without benefit are discarded, edits with positive but uncertain transfer are kept in external memory for retrieval, and edits with strong transfer are consolidated by context distillation, in which the policy reading the memory teaches the same policy without it.

Our contributions are threefold. (a) We separate help in context (read value) from improvement of the policy (learning value), and show that source benefit poorly predicts transfer while transfer is a cheap, useful proxy for learning value. (b) We propose MECA, which estimates each edit's transfer on query tasks disjoint from its source and uses this estimate, with its uncertainty and worst-query effect, both to reward memory editing and to select memories for consolidation. (c) On ALFWorld, WebShop, and Search-QA with three backbones, MECA improves over GiGPO by 11.6 points on average, outperforms controlled baselines that simply add memory, and retains 85% of its gain with all memory removed at inference; GiGPO trained on as many environment trajectories closes little of this gap, and variants that remove only transfer-based credit and selection remain well below it.

2 Problem Setup and Observations

Agent, memory, and objective. At iteration t, the agent consists of a policy π_{θ_t} and a memory store M_t. Each memory holds procedural guidance, when it applies, and pointers to its source experience. Retrieval gives task q a context $C_q = \mathcal{R}(q; M_t)$, fixed within an episode, and the policy acts by $\pi_{\theta_t}(\cdot \mid q, h_\ell, C_q)$ given history h_ℓ. Its success probability with context C is $V_\theta(q, C) = \mathbb{E}_{\tau \sim \pi_\theta(\cdot|q,C)}[W_q(\tau)]$, where $W_q(\tau) \in \{0, 1\}$ indicates success. Training updates the parameters and edits the store: an *edit* e, proposed from a source task, adds, revises, or deletes a memory, or is a no-op, and $C_q \oplus e$ is the context of q with e applied. The goal is high final success on held-out tasks, $J(\theta_T, M_T) = \mathbb{E}_q[V_{\theta_T}(q, \mathcal{R}(q; M_T))]$; $J(\theta_T, \varnothing)$ isolates what the parameters carry.

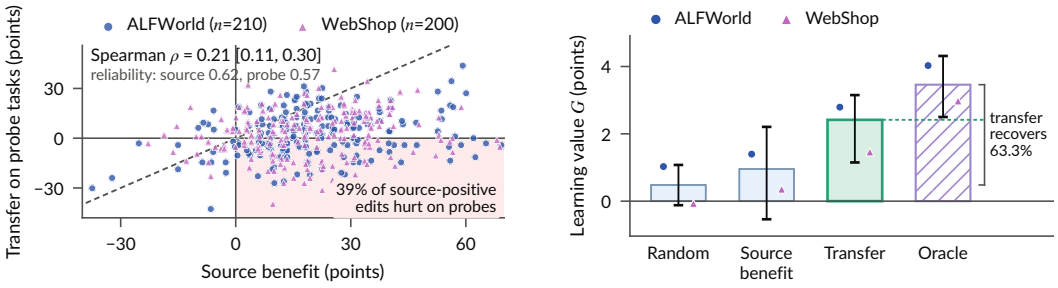
Read value, transfer, and learning value. The effect of e on q and its *read value* on tasks Q are

$$\Delta_t(e; q) = V_{\theta_t}(q, C_q \oplus e) - V_{\theta_t}(q, C_q), \qquad U_t(e; Q) = \frac{1}{|Q|} \sum_{q \in Q} \Delta_t(e; q). \tag{1}$$

Measured on fresh replays of its own source task, U_t is the edit's *source benefit*; measured on tasks disjoint from its source (tasks sharing an environment instance count as one source), it is the edit's *transfer*. *Learning value* instead asks what a memory does once learned: with θ^e the policy after consolidating e by context distillation (Section 3.3) and θ^0 the policy after an ordinary RL update on the same turns from the same checkpoint, with equal optimizer steps and tokens (Appendix G),

$$G_t(e) = \mathbb{E}_q \left[V_{\theta^e}(q, C_q) - V_{\theta^0}(q, C_q) \right], \tag{2}$$

with q from held-out *probe tasks* and e absent from both, so G_t measures learning, not reading.



(a) Source benefit vs. transfer

(b) Learning value by selection rule

Figure 2: **Source benefit misleads; transfer selects what is worth learning.** (a) One point per edit; dashed: $y = x$. (b) Top 25% of edits under each rule, relative to a matched update (0); error bars: 95% CI; markers: per-environment means; the oracle selects by learning value on a separate panel.

Observational study. We first study additions written by DeepSeek-V4-Flash (DeepSeek-AI et al., 2026), each from 8 rollouts of the base policy (Qwen2.5-3B-Instruct) on one source task, and evaluate them with that fixed policy and an empty store on same-family probe tasks (Appendix G).

2.1 Observation 1: Source benefit poorly predicts transfer

Across 410 edits, source benefit and probe transfer correlate at only $\rho = 0.21$ (95% CI [0.11, 0.30]; Figure 2a), although both are reliable (test–retest $\rho = 0.62$, split-half $\rho = 0.57$). 88% of edits help their source task, yet 39% of these hurt probe tasks: an edit explaining its source episode exploits non-recurring details, such as an object's location. Keeping the top 30% of edits by transfer on a separate query panel gives mean probe transfer 9.5 points, versus 5.8 by source benefit; a single disjoint query task, scored with as many rollouts as the source panel, gives 11.5, so disjointness alone accounts for this gain. **Implication:** MECA credits edits by disjoint-task transfer (Section 3.1).

2.2 Observation 2: Transfer selects memories worth consolidating

Reading and learning differ: across 120 edits, each consolidated in its own branch, transfer and learning value correlate at only $\rho = 0.32$, and 34% of positive-transfer edits yield no learning value ($G_t \leq 0$). Yet transfer allocates a limited consolidation budget well: consolidating the top 25% by transfer yields a mean learning value of 2.4 points, versus 0.5 for random and 1.0 for source-based selection, recovering 63.3% of the gain over random selection of an oracle that selects by measured learning value (3.5 points; Figure 2b). **Implication:** learning value needs a training branch per edit, whereas transfer comes free with the editing reward, so MECA uses transfer to choose what to consolidate (Section 3.3) and evaluates the consolidated policy without memory (Section 4.4).

3 MECA: Memory Evolution and Consolidation

MECA trains one policy to act, edit its memory, and consolidate selected memories (Figure 3): the agent proposes edits whose transfer we measure (Section 3.1), one RL update trains acting and editing (Section 3.2), and transfer routes each edit to be discarded, retrieved, or consolidated (Section 3.3).

Agent and memory. The same policy π_{θ_t} both acts and proposes edits from source trajectories. The store has two pools, $M_t = (M_t^{\text{ext}}, M_t^{\text{rev}})$: *external* memories are retrieved in training and deployment, whereas *review* memories hold the edits with the strongest transfer and are used only during training, as teacher context for consolidation and occasionally for rehearsal. Once consolidated, review memories need not be read, keeping the deployment context short and gains independent of retrieval (Section 4.4). Editing is initialized by supervised fine-tuning on demonstrations from an external labeler that sees only source experience, then trained online by RL (Appendix C).

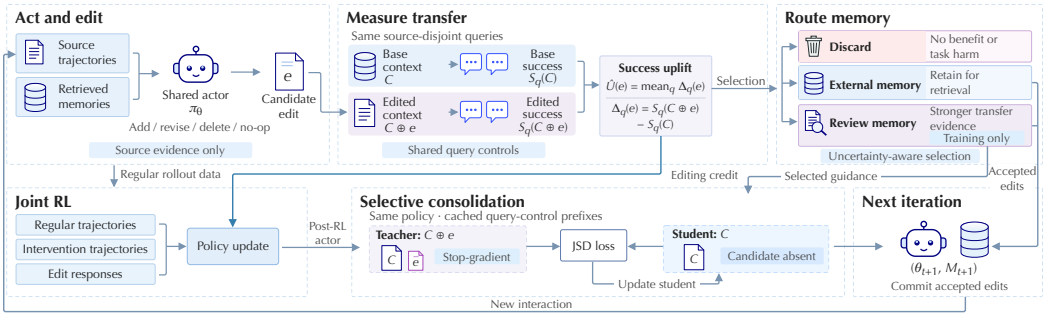


Figure 3: **One MECA iteration.** The actor proposes an edit e, whose transfer $\widehat{U}(e)$ is its success uplift on source-disjoint queries. Transfer serves as editing credit in joint RL and routes e to discard, external memory (retrieved at deployment), or review memory (training only). Review memories are distilled into the policy: a teacher reading $C \oplus e$ supervises a student reading C.

3.1 Memory evolution with transfer rewards

To reward memories that help beyond their source, we split each training batch, before any rollout, into source tasks S_t and query tasks Q_t; both provide trajectories for training to act, but only source trajectories are shown when editing. Policy and memory are frozen during collection.

Proposing edits. In each iteration, four of the eight source tasks, chosen before any rollout, are used for editing (Algorithm 1). For such a task s, a dossier D_s collects its sampled trajectories, their outcomes, and its retrieved memories. The agent proposes an edit $e \sim \pi_{\theta_t}(\cdot \mid D_s)$. Each edit cites trajectories from its dossier, and revisions and deletions may target only memories in the dossier.

Measuring transfer. Each candidate is applied, in isolation, to the context of every query task, giving $C_q \oplus e$: an addition inserts the new memory, a revision replaces its parent (or is inserted if the parent was not retrieved for q), and a deletion removes its parent. These interventions are temporary; the persistent store changes only at the end of the iteration. With $\widehat{V}_t^0(q)$ and $\widehat{V}_t^e(q)$ the empirical success rates on q under the base and edited contexts, we estimate the transfer $U_t(e; Q_t)$ of Eq. (1) as

$$\widehat{\Delta}_q(e) = \widehat{V}_t^e(q) - \widehat{V}_t^0(q), \qquad \widehat{U}(e) = \frac{1}{|Q_t|} \sum_{q \in Q_t} \widehat{\Delta}_q(e). \tag{3}$$

The editing reward is $r^{\text{edit}}(e) = \widehat{U}(e)$ for a valid edit, 0 for a no-op, and $-c_{\text{inv}}$ for an invalid response, with a fixed penalty $c_{\text{inv}} > 0$. All candidates share the same base-context query rollouts as controls, and the policy and other memories are fixed, so a change in query success can be attributed to the edit itself; the reward thus favors edits that help other tasks and penalizes edits that hurt them.

3.2 Learning to act and edit

One policy update trains acting and editing on three kinds of responses: *regular trajectories* (acting with the retrieved context C_q), *intervention trajectories* (acting on query tasks with the edited context $C_q \oplus e$), which teach the policy to use new memories and not only to write them, and *edit responses* (proposals generated from dossiers). Regular trajectories use GiGPO's episode-level and anchor-state advantages A^{reg} (Feng et al., 2025). An intervention trajectory with return $R^e_{q,k}$ is baselined by the mean base-context return $\overline{R}^0_q$ on the same query, and an edit, being a single response, uses its reward:

$$A^{\text{aug}}_{q,k} = R^e_{q,k} - \overline{R}^0_q, \qquad A^{\text{edit}}(e) = r^{\text{edit}}(e). \tag{4}$$

Centering on the base-context mean, rather than within the intervention group, preserves the signal that an edit improved success even when all of its intervention rollouts receive the same return.

A response token y_j of role r is generated under context $\widetilde{x}_j$ and has importance ratio w_j:

$$\widetilde{x}_j = \begin{cases} (q, h_\ell, C_q), & r = \text{reg}, \\ (q, h_\ell, C_q \oplus e), & r = \text{aug}, \\ D_s, & r = \text{edit}. \end{cases} \qquad w_j(\theta; \widetilde{x}_j) = \frac{\pi_\theta(y_j \mid \widetilde{x}_j, y_{<j})}{\pi_{\theta_{\text{old}}}(y_j \mid \widetilde{x}_j, y_{<j})}. \tag{5}$$

Each role r uses the clipped proximal policy optimization (PPO) loss (Schulman et al., 2017),

$$\mathcal{L}_r = -\frac{1}{|\mathcal{T}_r|} \sum_{j \in \mathcal{T}_r} \min\left\{ w_j(\theta; \widetilde{x}_j) A_j, \mathrm{clip}\big(w_j(\theta; \widetilde{x}_j), 1-\epsilon, 1+\epsilon\big) A_j \right\}, \tag{6}$$

where $\mathcal{T}_r$ contains the response tokens of role r and ϵ is the clipping radius. The joint objective is $\mathcal{L}_{\text{joint}} = \sum_r \lambda_r \mathcal{L}_r + \beta_{\text{KL}} \mathcal{L}_{\text{KL}} - \beta_H \mathcal{H}$, where $\mathcal{L}_{\text{KL}}$ is a Kullback–Leibler (KL) penalty toward a reference policy and $\mathcal{H}$ is the entropy; averaging per role gives acting and editing explicit weights λ_r; otherwise the far shorter edit responses would barely affect the gradient (Appendix C).

3.3 Consolidating transferable memories

Routing edits by transfer. Edits that do not help on average, or that sharply lower success on some query task, are discarded; among the rest, we separate edits whose benefit is reliably positive from those only positive on average. Let s_e be the standard error of $\widehat{U}(e)$, v_q the variance of $\widehat{\Delta}_q(e)$, and z a confidence multiplier. The lower bound $L_e = \widehat{U}(e) - z s_e$ measures reliable benefit, and $B_e = \min_{q \in Q_t}[\widehat{\Delta}_q(e) + z\sqrt{v_q}]$, an optimistic bound on the worst query effect, flags harm:

$$\text{route}(e) = \begin{cases} \text{DISCARD}, & \widehat{U}(e) \le 0 \text{ or } B_e \le -\rho, \\ \text{REVIEW}, & L_e \ge \eta \text{ and } e \text{ is not a deletion}, \\ \text{EXTERNAL}, & \text{otherwise}, \end{cases} \tag{7}$$

for thresholds η and ρ (cluster-robust s_e and v_q in Appendix B). Review-routed additions and revisions enter the training-only pool for consolidation; other accepted edits enter the external pool, and accepted deletions remove their targets. External memory helps as soon as an edit is committed, whereas consolidation costs distillation steps, so it is reserved for the most reliably transferable memories, which Observation 2 (Section 2.2) shows transfer can identify.

Distilling selected memories into the policy. Each review-routed edit is consolidated by context distillation (Snell et al., 2022; Agarwal et al., 2024): the policy reading the memory teaches the same policy without it, reusing query-task turns collected under the base context. For turn u with prompt x_u, response prefix $y_{u,<j}$, and edited prompt x_u^e containing the candidate, define

$$p_{u,j} = \pi_\vartheta(\cdot \mid x_u, y_{u,<j}; T), \qquad q_{u,j} = \mathrm{sg}[\pi_\vartheta(\cdot \mid x_u^e, y_{u,<j}; T)], \tag{8}$$

where ϑ, the distilled parameters, start from the post-RL policy, T is the distillation temperature (Hinton et al., 2015), and sg stops teacher gradients. With shared parameters, the teacher's only advantage is the memory; matching it teaches the student to act as if reading it. We minimize a generalized Jensen–Shannon divergence (JSD) (Lin, 1991; Agarwal et al., 2024),

$$\mathcal{L}_{\text{distill}} = \frac{1}{|\mathcal{B}|} \sum_{u \in \mathcal{B}} \frac{1}{L_u} \sum_{j=1}^{L_u} \left[(1-\alpha)\, \mathrm{KL}(p_{u,j} \| m_{u,j}) + \alpha\, \mathrm{KL}(q_{u,j} \| m_{u,j}) \right], \tag{9}$$

where $m_{u,j} = (1-\alpha)p_{u,j} + \alpha q_{u,j}$ is a mixture controlled by α, the batch $\mathcal{B}$ contains turn–candidate pairs, and L_u is the response length. Teacher distributions are recomputed from the current parameters at every step, while response prefixes are those collected before the RL update, so consolidation needs no extra environment interaction (compute cost in Section 4). Finally, accepted edits are committed to their pools, yielding (θ_{t+1}, M_{t+1}), from which the next iteration starts (Algorithm 1).

4 Experiments

We test three claims: MECA outperforms controlled baselines, including ones that simply add memory (Section 4.2); its gains come from crediting and selecting memories by transfer, not from the extra training this entails (Section 4.3); and transfer yields memories that help on tasks they were not written from, which consolidation turns into capability (Section 4.4).

4.1 Experimental setup

Benchmarks and models. We use ALFWorld (Shridhar et al., 2021) (success on `valid_seen`), WebShop (Yao et al., 2022) (task score and success), and the Search-QA suite (Feng et al.,

2025; Lu et al., 2026): exact match on NQ (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), PopQA (Mallen et al., 2023), HotpotQA (Yang et al., 2018), 2Wiki (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and Bamboogle (Press et al., 2023), and their average weighted by evaluation-set size. We train Qwen2.5-7B-Instruct, Qwen2.5-3B-Instruct, and Qwen3-1.7B (Yang et al., 2025b;a).

Training and evaluation. Every method trained in our pipeline runs 150 iterations of 16 tasks with 8 rollouts each (128 environment trajectories per iteration) and is evaluated at the final checkpoint. MECA also collects 64 intervention rollouts per iteration to measure transfer; consolidation reuses cached turns. We evaluate MECA with its external memory (editing disabled) and with *all memory removed*, which tests what the policy itself has learned. Comparisons are paired (shared seeds and evaluation tasks). Protocols, hyperparameters, and prompts are in Appendices E, C, and D.

Baselines. Table 1 compares MECA with our runs of memory-free RL, GRPO (Shao et al., 2024) and GiGPO (Feng et al., 2025), and with the results reported for self-distillation and experience-based methods: SDAR (Lu et al., 2026), OPID (Yang et al., 2026a), SEED (Wu et al., 2026), UCOB (Tu et al., 2026), GRSD (Zheng et al., 2026), and AgentOPSD (Wang et al., 2026). Published results retain each paper's protocol, with Search-QA averages recomputed using common weights. Our conclusions rest on the controlled comparison of Table 2, where we train every method, including GRPO and GiGPO, in our pipeline with the same base models, prompt format, context limits, and training schedule. Besides GRPO and GiGPO, it includes two GiGPO controls matched to MECA's environment trajectories: **GiGPO, 12 rollouts**, which samples 12 rollouts per task (192 trajectories per iteration), and **GiGPO, 225 iterations**, which trains for 225 iterations of 128 trajectories; **GiGPO + labeler SFT**, first fine-tuned on the demonstrations of the cold-start labeler (DeepSeek-V4-Flash); **GiGPO + static memory**, which retrieves memories written once by the labeler; and **GiGPO + unfiltered memory**, which proposes edits with MECA's prompt, dossiers, and cold start from 2,048 labeler demonstrations, but commits every valid edit and gives its editor no reward, routing, or consolidation.

4.2 Main results

MECA is competitive with published methods. In Table 1, MECA has the highest size-weighted Search-QA average on every backbone and ALFWorld success on both Qwen2.5 models. It is not uniformly best: SEED and UCOB report higher WebShop success on Qwen2.5-3B, several methods higher WebShop scores on both smaller backbones, and GRSD and SEED lead slightly on Qwen3-1.7B ALFWorld.

MECA outperforms every controlled baseline. In Table 2, MECA leads every baseline in all nine backbone–benchmark pairs, including the strongest, GiGPO + unfiltered memory. Averaged over backbones, it improves over GiGPO by 12.5, 18.7, and 3.4 points on ALFWorld, WebShop, and Search-QA, most on the smallest backbone (WebShop success from 46.9 to 81.3 on Qwen3-1.7B).

Memory access and extra supervision do not explain the gains. Static labeler memory, unfiltered self-written memory, and labeler SFT improve GiGPO by only 4.5, 5.3, and 3.4 points on average. Unfiltered memory shares MECA's demonstrations and edit format, yet MECA leads it by 6.3 points: what matters is which edits are kept and learned from.

The gains persist without memory. MECA still leads GiGPO by 9.8 points on average and retains 85% of its gain, while GiGPO + static memory keeps only 1.0 of its 4.5 points (Appendix Table 4).

4.3 What drives the gains

Figure 4 replaces one component at a time on Qwen2.5-3B-Instruct and adds controls that match MECA's environment trajectories or, within its pipeline, its intervention rollouts and distillation steps. We focus on memory-free success, which measures what the policy has learned.

More trajectories do not close the gap. Giving GiGPO as many environment trajectories as MECA, as 12 rollouts per task or as 225 iterations, improves it by only 0.3 and 0.9 points, and MECA without memory still leads the stronger control by 4.1 points (Figure 4a).

Transfer is the editing signal that matters. *No transfer credit* keeps MECA's full pipeline, including intervention rollouts, all losses, and distillation steps, but shuffles editing rewards across candidates

Method	ALFWorld							WebShop		Search-QA							
	Pick	Look	Clean	Heat	Cool	Pick2	Succ	Score	Succ	NQ	Triv	Pop	Hotp	2Wk	MuS	Bam	Avg
Qwen2.5-7B-Instruct																	
GRPO Shao et al. (2024)‡	92.9	38.9	80.8	78.6	94.7	73.9	78.1	86.4	71.9	42.5	57.8	45.8	43.1	43.2	18.0	36.0	45.9
GiGPO Feng et al. (2025)‡	100.0	66.7	95.7	70.0	70.8	95.2	87.5	85.9	72.7	45.8	64.9	44.5	40.5	43.5	20.6	64.0	47.2
SDAR Lu et al. (2026)	94.7	75.0	100.0	86.7	68.2	78.9	85.9	89.4	82.8	46.3	63.5	48.2	43.8	48.4	19.6	73.0	49.6
OPID Yang et al. (2026a)	100.0	81.8	97.1	100.0	80.8	80.0	90.0	85.3	79.7	48.8	65.6	46.8	46.1	42.7	21.7	72.6	48.8
SEED Wu et al. (2026)	100.0	100.0	96.3	80.0	100.0	100.0	96.1	89.7	78.1	47.0	64.9	47.8	45.2	45.3	20.1	70.2	49.3
UCOB Tu et al. (2026)	93.8	90.9	94.1	95.8	92.9	89.5	93.0	91.8	85.9	46.4	65.6	46.7	45.7	42.9	19.2	70.6	48.5
GRSD Zheng et al. (2026)	100.0	100.0	100.0	93.4	66.7	82.4	92.2	91.6	82.8	47.5	65.6	49.6	44.5	47.1	20.7	72.1	50.3
AgentOPSD Wang et al. (2026)	91.2	87.5	100.0	90.5	75.0	84.2	89.1	90.2	79.7	47.5	64.9	46.8	45.8	45.3	20.9	70.2	49.1
MECA	100.0	100.0	100.0	100.0	93.5	95.8	97.7	93.4	86.7	49.6	65.5	49.7	43.3	48.3	22.0	72.8	50.7
Qwen2.5-3B-Instruct																	
GRPO Shao et al. (2024)‡	86.5	64.3	85.0	76.9	62.1	60.0	74.2	83.8	62.5	38.7	60.1	41.0	35.7	34.8	16.5	27.2	41.6
GiGPO Feng et al. (2025)‡	96.9	68.8	84.6	100.0	73.9	90.0	85.9	82.9	69.5	42.9	60.0	46.2	37.2	35.8	14.6	53.6	43.7
SDAR Lu et al. (2026)	97.1	62.5	100.0	61.9	75.0	84.2	84.4	85.0	68.0	44.8	58.1	44.3	38.6	36.2	15.7	66.1	43.3
OPID Yang et al. (2026a)	92.7	100.0	88.9	70.0	84.2	70.0	84.3	85.0	74.2	45.9	61.4	45.7	40.7	38.8	16.4	66.1	45.4
SEED Wu et al. (2026)	100.0	100.0	100.0	100.0	70.6	80.0	91.8	88.5	78.9	44.3	61.9	47.2	41.0	42.0	16.9	67.7	46.7
UCOB Tu et al. (2026)	93.8	93.3	95.8	94.1	91.0	78.9	92.2	84.5	78.1	45.0	61.4	47.0	37.2	35.8	12.5	66.2	44.3
GRSD Zheng et al. (2026)	100.0	81.3	91.3	88.2	70.8	85.7	86.7	86.9	76.5	49.4	59.9	42.9	43.8	42.8	20.0	16.2	46.0
AgentOPSD Wang et al. (2026)	100.0	68.8	86.4	66.7	84.6	73.7	84.4	90.4	69.5	44.9	61.9	48.0	40.9	41.6	14.4	68.1	46.7
MECA	100.0	83.3	100.0	95.2	80.0	95.7	94.5	88.0	77.3	49.9	62.8	47.3	40.5	43.5	17.9	65.6	47.6
Qwen3-1.7B																	
GRPO Shao et al. (2024)‡	64.9	22.2	48.3	53.8	35.0	30.0	46.9	71.1	41.4	38.9	58.4	43.5	34.0	30.3	12.6	63.2	40.5
GiGPO Feng et al. (2025)‡	84.2	68.8	64.7	54.5	68.0	76.2	72.7	70.0	46.9	41.2	59.4	45.1	37.0	30.9	12.2	66.4	41.9
SDAR Lu et al. (2026)	73.5	25.0	76.9	33.3	40.0	36.8	53.9	76.8	58.6	39.7	58.9	45.3	35.9	35.5	12.6	65.3	42.7
OPID Yang et al. (2026a)	65.9	72.7	66.7	40.0	63.2	45.0	58.9	79.6	64.8	38.1	58.1	43.4	35.5	31.7	11.7	64.5	40.8
SEED Wu et al. (2026)	97.6	100.0	100.0	80.0	84.2	90.0	92.0	87.1	77.3	42.0	58.9	47.1	36.9	35.2	10.6	64.9	43.3
UCOB Tu et al. (2026)	96.2	85.7	95.5	84.2	81.5	88.8	89.1	91.5	79.7	44.1	59.6	45.7	36.8	31.9	12.2	64.5	42.5
GRSD Zheng et al. (2026)	92.5	100.0	100.0	91.3	81.3	88.2	92.2	84.5	66.4	41.8	60.0	45.8	38.1	36.4	12.2	66.1	43.7
MECA	96.0	75.0	100.0	100.0	78.9	95.2	91.4	83.7	81.3	46.7	64.9	47.0	40.4	31.8	14.3	67.2	44.8

Table 1: **Main results** (%). ALFWorld: success per task family and overall (Succ); WebShop: score and success; Search-QA: exact match per dataset and their size-weighted average (Avg), recomputed for all methods (Appendix E). Shaded: aggregate metrics. ‡Our runs in the pipeline of Table 2; other baseline scores, except Search-QA Avg, are as reported in the cited papers.

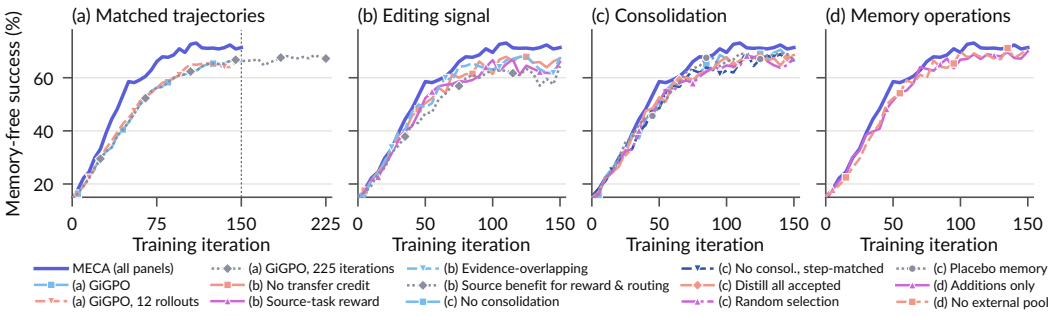


Figure 4: **Ablations and budget-matched controls** (Qwen2.5-3B-Instruct; memory-free success averaged over ALFWorld, WebShop, and Search-QA). (a) GiGPO matched to MECA's environment trajectories, from 12 rollouts per task or 225 iterations (vertical line: iteration 150). (b) *No transfer credit* shuffles editing rewards and routes at random; the other variants score edits on tasks including their sources. (c) *Random selection* and *placebo memory* use MECA's distillation budget; the *no consolidation* variants skip distillation (the step-matched one replaces it with ordinary steps); *distill all accepted* uses a larger budget. Final values with and without memory: Appendix Table 12.

and routes at random; it trails MECA by 4.0 points and leads GiGPO by only 1.0 (Figure 4b). The gains thus require selecting edits by transfer, not the extra training this entails. Rewarding edits by source benefit, estimated with as many rollouts as transfer with routing unchanged, lowers memory-free success by 6.0, 7.3, and 2.9 points on ALFWorld, WebShop, and Search-QA, respectively.

Method	Qwen2.5-7B-Instruct			Qwen2.5-3B-Instruct			Qwen3-1.7B		
	ALFW	WebS	SQA	ALFW	WebS	SQA	ALFW	WebS	SQA
Memory-free RL									
GRPO	78.1	71.9	45.9	74.2	62.5	41.6	46.9	41.4	40.5
GiGPO	87.5	72.7	47.2	85.9	69.5	43.7	72.7	46.9	41.9
GiGPO, 12 rollouts	91.4	74.0	47.5	86.5	70.6	42.8	80.5	46.9	41.2
GiGPO, 225 iterations	91.9	74.5	47.6	87.2	71.4	43.1	81.0	47.7	41.4
Memory and extra supervision									
GiGPO + labeler SFT	92.2	76.3	47.8	88.0	71.9	43.5	82.6	54.7	41.7
GiGPO + static memory	92.7	77.6	48.2	88.5	72.9	43.9	84.1	58.1	42.1
GiGPO + unfiltered memory	93.5	78.6	48.4	89.1	73.4	44.2	84.9	61.2	42.4
MECA (memory removed)	95.8	84.1	49.6	92.7	75.0	46.4	89.8	78.9	43.9
MECA	**97.7**	**86.7**	**50.7**	**94.5**	**77.3**	**47.6**	**91.4**	**81.3**	**44.8**

Table 2: **Controlled comparison.** All methods, including GRPO and GiGPO, are trained in our pipeline with the same base model, prompt format, context limits, and evaluation protocol; no row is copied from a paper. Unless noted, each method runs 150 iterations of 16 tasks $\times$ 8 rollouts. MECA adds 64 intervention rollouts per iteration; GiGPO with *12 rollouts* per task or *225 iterations* matches its environment trajectories; *labeler SFT* fine-tunes on cold-start labeler demonstrations before GiGPO; *memory removed* evaluates MECA without memory. ALFW: ALFWorld success (`valid_seen`); WebS: WebShop success; SQA: Search-QA size-weighted exact match (%).

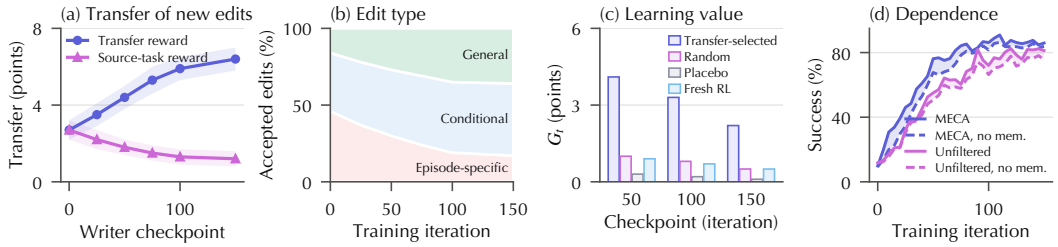


Figure 5: **How transfer shapes memory and consolidation** (Qwen2.5-3B-Instruct, ALFWorld and WebShop). (a) Transfer per edit slot of edits that saved checkpoints write on held-out dossiers, read by the base policy; shading: std over seeds. (b) Accepted edits by type. (c) Learning value of candidates from MECA checkpoints, relative to a matched ordinary update. (d) Dependence on memory (shaded): success with minus without memory; *Unfiltered*: GiGPO + unfiltered memory.

Scoring edits on queries that also supply the induction evidence, as in SAPO (Zhang et al., 2026c), recovers only 31.5% of the mean gap across benchmarks; using source benefit for routing as well costs a further 1.6 points without memory.

Consolidation needs selection. Disabling consolidation lowers memory-free success by 4.9 points on average (Figure 4c). Replacing each distillation step with an ordinary step recovers only 0.2 of these points; distilling random edits or a placebo memory under the same budget recovers 0.8 and 0.3, and distilling every accepted addition and revision, with a larger budget, 2.0. As teacher and student share parameters, distilling with no memory would give zero loss; the placebo keeps the procedure and budget but uses an unrelated memory of the same length, so its small effect shows the benefit comes from the content of the selected memories.

Other components. Point estimates also favor the remaining components: allowing only additions lowers memory-free success by 1.5 points, and removing the external pool for edits with positive but uncertain transfer, the editing loss, or the intervention loss lowers memory-free success by 0.4, 3.1, and 1.8 points on average. Without cold start, MECA still leads GiGPO by 3.3 points without memory, and a separate editor model brings no gain (Appendix Table 12).

4.4 How memory becomes capability

Transfer reward teaches the agent to write memories that transfer. Saved checkpoints write edits for 200 fixed held-out dossiers, which a common frozen reader uses on independent probes. Under transfer reward, transfer per edit slot rises from 2.7 to 6.4 points over training, while under source-task reward it falls to 1.2 (Figure 5a); the ordering holds under every reader (Appendix Table 9). The content changes accordingly: episode-specific edits, such as an object's location, fall from 46% to 17.0% of accepted edits, and general procedures rise to 36.0% (Figure 5b), at similar edit length. The online estimate is noisy but informative (Spearman 0.68 with independent probes), and selected edits remain strongly positive when re-scored on fresh panels (Appendix G).

Transfer selects memories worth consolidating. Repeating Observation 2 with MECA's own edits at iterations 50, 100, and 150, transfer-selected candidates have a mean learning value of 3.2 points, versus 0.8 for random candidates, 0.2 for a placebo, and 0.7 for a compute-matched fresh-rollout RL update (Figure 5c). Relative to a matched update, consolidation raises success without the memory (56.3 vs. 53.1) and slightly raises it with the memory (63.0 vs. 62.3), so the policy learns the behavior rather than losing the ability to use the memory; masking paraphrases of the memory changes this by less than 0.6 points, and success on other task families is unaffected (Appendix Table 8).

Dependence on memory shrinks. For MECA, the gap between success with and without memory peaks at 11.7 points early and narrows to 2.1, whereas for unfiltered memory it widens to 5.6 (Figure 5d); gains persist at least 50 iterations after consolidation stops (Appendix Figure 7b).

Robustness and cost. Training and evaluating with a dense retriever instead of the lexical one changes MECA's success by $+0.4$ points with memory and $+0.3$ without (Appendix I). Editing, intervention rollouts, and distillation raise wall-clock time per iteration from GiGPO's 183 to 245 seconds, and MECA reaches GiGPO's final ALFWorld success with 50% of GiGPO's environment trajectories (Appendix Figure 6). Example memories are in Appendix J.

5 Related work

Learning to construct and use memory. Reflexion, ExpeL, and Agent Workflow Memory turn interaction experience into reflections, insights, and reusable workflows (Shinn et al., 2023; Zhao et al., 2024; Wang et al., 2024b). Memory-R1 and UMA train memory operations from task feedback, MemRL learns retrieval utilities with frozen weights, and Skill1 co-trains skill selection, execution, and generation (Yan et al., 2025; Zhang et al., 2026a;b; Shi et al., 2026). MECA also learns to edit memory, but credits each edit by its effect on tasks disjoint from its source.

Feedback for memory evolution. Which tasks evaluate an edit determines what its reward measures. SAPO evaluates skill utility on queries that also supply the induction evidence (Zhang et al., 2026c); SkillCAT uses source-task replays (Chen et al., 2026b); SkillOpt, EvoSkill, and GEPA use held-out selection to optimize textual artifacts without training the policy (Yang et al., 2026b; Alzubi et al., 2026; Agrawal et al., 2025). MECA measures transfer on disjoint query tasks during joint policy training and reuses it to choose what to consolidate.

Consolidating guidance into the policy. Context distillation moves behavior elicited by extra context into weights (Snell et al., 2022; Agarwal et al., 2024). SDAR gates skill supervision, OPID and SEED distill evolving hindsight, and GRSD compares group reflections (Lu et al., 2026; Yang et al., 2026a; Wu et al., 2026; Zheng et al., 2026). UCOB, closest to our work, combines bidirectional distillation, utility-aware memory, and writer training (Tu et al., 2026); SkillC, PEAM, and Dual-Layer Agentic Memory study skill retirement, consolidation opportunities, and validated eviction (Lin et al., 2026; Guo et al., 2026; Li et al., 2026). As internalizing skills is hard (Chen et al., 2026a), MECA selects what to consolidate by transfer and evaluates the consolidated policy without memory.

6 Conclusion

A memory becomes capability when it transfers to tasks it was not written from. MECA uses this criterion to reward memory edits and select what to consolidate; it outperforms controlled baselines matched on environment trajectories, writes increasingly transferable memories, and retains most gains without memory at inference.

REPRODUCIBILITY STATEMENT

The method equations specify editing rewards, joint policy learning, and selective distillation. Appendix A gives the complete iteration; Appendices B and C specify routing, initialization, retrieval, and optimization. Data separation, uncertainty estimation, and controlled comparisons are described in Appendices E, G, and H.

AI USE STATEMENT

Generative AI assisted with code implementation and polishing the wording in the paper.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/5be69a584901a26c521c2b51e40a4c20-Abstract-Conference.html.

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*, 2025. URL https://arxiv.org/abs/2507.19457.

Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. EvoSkill: Automated skill discovery for multi-agent systems. *arXiv preprint arXiv:2603.02766*, 2026. URL https://arxiv.org/abs/2603.02766.

Lawrence D. Brown, T. Tony Cai, and Anirban DasGupta. Interval estimation for a binomial proportion. *Statistical Science*, 16(2):101–133, 2001. doi: 10.1214/ss/1009213286. URL https://doi.org/10.1214/ss/1009213286.

A. Colin Cameron and Douglas L. Miller. A practitioner's guide to cluster-robust inference. *Journal of Human Resources*, 50(2):317–372, 2015. doi: 10.3368/jhr.50.2.317. URL https://doi.org/10.3368/jhr.50.2.317.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 2318–2335, 2024. doi: 10.18653/v1/2024.findings-acl.137. URL https://aclanthology.org/2024.findings-acl.137/.

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*, 2026a. URL https://arxiv.org/abs/2606.04703.

Kunfeng Chen, Qihuang Zhong, Juhua Liu, and Bo Du. SkillCAT: Contrastive, assessment-augmented and topology-aware skill self-evolution for LLM agents. *arXiv preprint arXiv:2606.13317*, 2026b. URL https://arxiv.org/abs/2606.13317.

Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960. doi: 10.1177/001316446002000104. URL https://doi.org/10.1177/001316446002000104.

DeepSeek-AI et al. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026. URL https://arxiv.org/abs/2606.19348.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. *arXiv preprint arXiv:2505.10978*, 2025. URL https://arxiv.org/abs/2505.10978.

Yuchen Guo, Junli Gong, Weicheng Wang, Hongmin Cai, Yiu-ming Cheung, and Weifeng Su. PEAM: Parametric embodied agent memory through contrastive internalization of experience in Minecraft. *arXiv preprint arXiv:2605.27762*, 2026. URL https://arxiv.org/abs/2605.27762.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. URL https://arxiv.org/abs/1503.02531.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, 2020. doi: 10.18653/v1/2020.coling-main.580. URL https://aclanthology.org/2020.coling-main.580/.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, 2017. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147/.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

Wenzhi Li, Dong Nie, Rui Lan, Tongtong Lyu, Peiyao Wang, Lingzi Hong, Weihang Pan, Binbin Lin, Boyuan Pan, and Yao Hu. Dual-layer agentic memory with fast write routing and slow consolidation. *arXiv preprint arXiv:2608.22215*, 2026. URL https://arxiv.org/abs/2608.22215.

Hongxiang Lin, Zhirui Kuai, Erpeng Xue, and Lei Wang. SkillC: Learning autonomous skill internalization in LLM agents via contrastive credit assignment. *arXiv preprint arXiv:2605.27899*, 2026. URL https://arxiv.org/abs/2605.27899.

Jianhua Lin. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991. doi: 10.1109/18.61115. URL https://doi.org/10.1109/18.61115.

Zhengxi Lu, Zhiyuan Yao, Zhuowen Han, Zi-Han Wang, Jinyang Wu, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Self-distilled agentic reinforcement learning. *arXiv preprint arXiv:2605.15155*, 2026. URL https://arxiv.org/abs/2605.15155.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9802–9822, 2023. doi: 10.18653/v1/2023.acl-long.546. URL https://aclanthology.org/2023.acl-long.546/.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, 2023. doi: 10.18653/v1/2023.findings-emnlp.378. URL https://aclanthology.org/2023.findings-emnlp.378/.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

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.

Yaorui Shi, Yuxin Chen, Zhengxi Lu, Yuchun Miao, Shugui Liu, Qi Gu, Xunliang Cai, Xiang Wang, and An Zhang. Skill1: Unified evolution of skill-augmented agents via reinforcement learning. *arXiv preprint arXiv:2605.06130*, 2026. URL https://arxiv.org/abs/2605.06130.

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

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

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

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022. doi: 10.1162/tacl_a_00475. URL https://aclanthology.org/2022.tacl-1.31/.

Songjun Tu, Chengdong Xu, Qichao Zhang, Yiwen Ma, Yaocheng Zhang, Linjing Li, Dong Li, Xiangyuan Lan, and Dongbin Zhao. UCOB: Learning to utilize and evolve agentic skills via credit-aware on-policy bidirectional self-distillation. *arXiv preprint arXiv:2606.29502*, 2026. URL https://arxiv.org/abs/2606.29502.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024a. ISSN 2835-8856. URL https://arxiv.org/abs/2305.16291.

Zi-Han Wang, Zhengxi Lu, Zhiyuan Yao, Jinyang Wu, Jie Wu, Zhengzhou Cai, Yueqing Sun, Ziang Ye, Linji Hao, Qi Gu, Xunliang Cai, Yongliang Shen, and Yujiu Yang. AgentOPSD: Recursive self-distillation for agentic reinforcement learning. *arXiv preprint arXiv:2608.05987*, 2026. URL https://arxiv.org/abs/2608.05987.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024b. URL https://arxiv.org/abs/2409.07429.

Jinyang Wu, Shuo Yang, Zhengxi Lu, Fan Zhang, Yuhao Shen, Lang Feng, Haoran Luo, Zheng Lian, Shuai Zhang, Zhengqi Wen, and Jianhua Tao. SEED: Self-evolving on-policy distillation for agentic reinforcement learning. *arXiv preprint arXiv:2607.14777*, 2026. URL https://arxiv.org/abs/2607.14777.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schütze, Volker Tresp, and Yunpu Ma. Memory-R1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *arXiv preprint arXiv:2508.19828*, 2025. URL https://arxiv.org/abs/2508.19828.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a. URL https://arxiv.org/abs/2505.09388.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025b. URL https://arxiv.org/abs/2412.15115.

Shuo Yang, Jinyang Wu, Zhengxi Lu, Yuhao Shen, Fan Zhang, Lang Feng, Shuai Zhang, Haoran Luo, Zheng Lian, Zhengqi Wen, and Jianhua Tao. OPID: On-policy skill distillation for agentic reinforcement learning. *arXiv preprint arXiv:2606.26790*, 2026a. URL https://arxiv.org/abs/2606.26790.

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, Kai Qiu, Yuqing Yang, Dongdong Chen, Xue Yang, and Chong Luo. SkillOpt: Executive strategy for self-evolving agent skills. *arXiv preprint arXiv:2605.23904*, 2026b. URL https://arxiv.org/abs/2605.23904.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, 2018. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/82ad13ec01f9fe44c01cb91814fd7b8c-Abstract-Conference.html.

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

Kehao Zhang, Shangtong Gui, Sheng Yang, Wei Chen, and Yang Feng. Learning to remember: End-to-end training of memory agents for long-context reasoning. *arXiv preprint arXiv:2602.18493*, 2026a. URL https://arxiv.org/abs/2602.18493.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Zhuo Li, Yujie Zheng, Weinan Zhang, Ying Wen, Zhiyu Li, Feiyu Xiong, Yutao Qi, Bo Tang, and Muning Wen. MemRL: Self-evolving agents via runtime reinforcement learning on episodic memory. *arXiv preprint arXiv:2601.03192*, 2026b. URL https://arxiv.org/abs/2601.03192.

Zhiwei Zhang, Yudi Lin, Nikki Lijing Kuang, Linlin Wu, Xiaomin Li, Songtao Liu, and Fenglong Ma. Co-evolving skill generation and policy optimization. *arXiv preprint arXiv:2606.08755*, 2026c. URL https://arxiv.org/abs/2606.08755.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024. doi: 10.1609/aaai.v38i17.29936. URL https://ojs.aaai.org/index.php/AAAI/article/view/29936.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, Alban Desmaison, Can Balioglu, Pritam Damania, Bernard Nguyen, Geeta Chauhan, Yuchen Hao, Ajit Mathews, and Shen Li. PyTorch FSDP: Experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023. URL https://arxiv.org/abs/2304.11277.

Binbin Zheng, Zijun Xie, Guanqun Zhao, Enlei Gong, Xing Ma, Xiaoliang Fu, and Zeyu Chen. Group-reflective self-distillation for agentic reinforcement learning. *arXiv preprint arXiv:2607.28076*, 2026. URL https://arxiv.org/abs/2607.28076.

A Complete training iteration

Algorithm 1 specifies one iteration; accepted edits are committed only after the policy update. If a batch cannot be partitioned into sources and queries, editing is skipped and the 64 intervention trajectories use the base contexts.

Algorithm 1 One MECA training iteration

Require: policy π_{θ_t}, store $M_t = (M_t^{\text{ext}}, M_t^{\text{rev}})$, 16 training tasks
Ensure: policy $\pi_{\theta_{t+1}}$, store M_{t+1}

Collection (policy and store frozen)
Split the tasks into sources S_t and queries Q_t with $|S_t| = |Q_t| = 8$; designate four editing sources $S_t^{\text{e}} \subset S_t$ ▷ from task metadata, seed, and iteration only, before any rollout
for all tasks $x \in S_t \cup Q_t$ **do**
  Retrieve one immutable context C_x from M_t and roll out 8 regular trajectories
end for ▷ 64 source and 64 query-control trajectories
for all sources $s \in S_t^{\text{e}}$ **do**
  Build the dossier D_s and propose an edit $e_s \sim \pi_{\theta_t}(\cdot \mid D_s)$
  for all queries $q \in Q_t$ **do**
    Roll out 2 intervention trajectories under $C_q \oplus e_s$ ▷ C_q if e_s is invalid or a no-op
  end for
  Compute $\widehat{U}(e_s)$ and $r^{\text{edit}}(e_s)$ (Eq. 3) and $\text{route}(e_s)$ (Eq. 7)
end for
Resolve conflicts among accepted edits; keep every edit response for learning
Joint RL
Compute A^{reg}, A^{aug}, and A^{edit} (Eq. 4)
Update θ_t by one clipped step on $\mathcal{L}_{\text{joint}}$ with per-role averages (Eq. 6), giving ϑ
Consolidation
if some edit is routed to review **then**
  Pair each review edit with cached query-control turns
  Run one distillation epoch on $\mathcal{L}_{\text{distill}}$ (Eq. 9) with the live optimizer ▷ teacher gradients stopped
end if
$\theta_{t+1} \leftarrow \vartheta$
Memory update
Commit accepted edits to M_t^{ext} or M_t^{rev} and apply accepted deletions, giving M_{t+1}
return $\pi_{\theta_{t+1}}$, M_{t+1}

B Exact uncertainty adjustment

This appendix specifies the uncertainty terms of the routing rule (Equation (7)). Write $n = 8$ for the query count, $d_q = \widehat{\Delta}_q(e)$, and $\bar{d} = \widehat{U}(e)$. For condition $c \in \{0, e\}$ with K_c trials and S_q^c successes, use binomial posterior moments under the Jeffreys prior (Brown et al., 2001) to define the smoothed rate and sampling-variance floor

$$\widetilde{p}_q^c = \frac{S_q^c + 1/2}{K_c + 1}, \qquad v_q = \sum_{c \in \{0,e\}} \frac{\widetilde{p}_q^c(1 - \widetilde{p}_q^c)}{K_c + 2}, \quad K_0 = 8, \quad K_e = 2. \tag{10}$$

Let $\mathcal{C}_1, \ldots, \mathcal{C}_g$ partition queries by exclusion identity. Combining the intercept-only cluster-robust variance (Cameron & Miller, 2015) with this floor gives the aggregate scale

$$v_{\text{between}} = \begin{cases} \dfrac{g}{g-1}\dfrac{1}{n^2}\displaystyle\sum_{c=1}^{g}\left[\sum_{q \in \mathcal{C}_c}(d_q - \bar{d})\right]^2, & g > 1, \\ 1, & g = 1, \end{cases} \tag{11}$$

$$s_e^2 = \max\left\{ v_{\text{between}}, \frac{1}{n^2}\sum_q v_q \right\}. \tag{12}$$

The rule then uses $L_e = \bar{d} - z s_e$ and $B_e = \min_q(d_q + z\sqrt{v_q})$; the regression veto thus uses the *upper* task estimate, so a negative point estimate alone does not trigger it. This heuristic addresses limited

replication; it is not a calibrated simultaneous bound, multiple-testing correction, or population guarantee. Independent probes and fresh panels assess routing quality and selection optimism (Table 7).

Interpretation of the mean. For a candidate chosen without query outcomes, a fixed panel, and unbiased condition means, linearity gives $\mathbb{E}[\widehat{U}(e)] = U_t(e; Q_t)$, without requiring equal sample counts or independence across candidates. Shared controls correlate candidate estimates, and selecting on them makes accepted-candidate estimates optimistic, so admission quality is evaluated on an independent panel; the identity does not guarantee policy improvement.

C Memory representation and implementation defaults

Cold-start initialization. An external labeler writes structured edit demonstrations from source trajectories of the pretrained actor under a frozen initial store; it sees no query outcomes and does not modify the store. Valid demonstrations, including no-ops (Appendix D), form $\mathcal{D}_{\text{cs}}$, and supervised initialization minimizes

$$\mathcal{L}_{\text{cs}}(\theta) = -\frac{1}{|\mathcal{D}_{\text{cs}}|} \sum_{(D,\tilde{e})\in\mathcal{D}_{\text{cs}}} \log \pi_\theta(\tilde{e} \mid D) \tag{13}$$

over the response and end-of-sequence tokens only.

Regular execution advantages. Regular trajectories use GiGPO's estimator (Feng et al., 2025), $A^{\text{reg}} = A^{\text{episode}} + \omega A^{\text{anchor}}$: the episode term compares returns within a task's rollout group, and the anchor term compares discounted returns of actions from the same environment state, both over regular trajectories only. Intervention trajectories use the query-control baseline (Equation (4)) without group centering or standardization. Execution returns include invalid-action costs; editing credit uses exact task success.

Memory records. Records store text, environment/task-family scope, applicability terms and anti-cues, source provenance, parent/concept identifiers, and residency, as in this WebShop record (bookkeeping fields omitted):


```
{"memory_id": "v8-memory_12190213f3c73dca",
 "text": "When a product page shows size options (e.g., 40x40x40cm, 110x40x40cm), click the
    exact size option that matches the task requirement before clicking 'Buy Now'.",
 "memory_type": "procedure", "concept_key": "webshop_select_size_option_before_buy", "
    task_families": ["webshop"],
 "applicability": {"anti_cues": [],
   "description": "When on a product detail page that offers size options and the task
     specifies a particular size, select the matching size option before purchasing.",
   "goal_terms": ["size", "buy now"], "state_cues": ["product page with size options", "task
     specifies a size"]},
 "residency": "external", "created_step": 8, "source_candidate_id": "v8-
    candidate_acd2ce0f3e656b3a"}
```


Citation and schema checks do not verify semantic entailment by the trajectories; behavioral utility is distinct from factual truth.

Retrieval. Eligible records are filtered by environment, status, and anti-cues. For token sets $T(q)$ and $T(m)$ formed from the query and memory text/applicability fields, respectively, the lexical score is

$$s(m,q) = \frac{|T(q)\cap T(m)|}{\sqrt{|T(q)|\,|T(m)|}} + 0.05\,\mathbb{I}[\text{task-family match}]. \tag{14}$$

Records are sorted by score and stable identifier, then packed into separate external and review budgets. During training, review exposure is sampled once per task group by a deterministic hash of run seed, group identity, actor version, and store revision, so all replicas receive the same context. Retriever token limits are whitespace-word estimates (actor prompt limits use the tokenizer) and do not cap the number of stored records.

Distillation batching. At most four edits contribute 16 turns each, filling at most one 64-turn minibatch, so distillation takes at most one step of the actor's optimizer (advancing its scheduler) per iteration. Teacher logits are detached and recomputed at current parameters for each microbatch; the loss averages active turns, excluding padding.

Setting	ALFWorld	WebShop
Base models	Qwen2.5-3B/7B-Instruct, Qwen3-1.7B	
Tasks / iteration (source / query)	16 (8 / 8)	
Regular trajectories / iteration	16 tasks $\times$ 8 rollouts $=$ 128	
Intervention trajectories / iteration (MECA only)	4 edits $\times$ 8 queries $\times$ 2 rollouts $=$ 64	
Maximum environment actions	50	15
Recent-history length	2	2
Actor prompt / response tokens	2048 / 512	4096 / 512
Editor prompt / response tokens	4096 / 512	
Source turns retained / trajectory	At most 8	
New memory text limit	128 words	
External retrieval: records / approximate tokens	3 / 384	
Review retrieval: records / approximate tokens	1 / 128	
Review exposure probability	0.125 per group	
Learning rate / PPO clip	10^{-6} / 0.2	
PPO epochs / ordinary optimizer steps per iteration	1 / 1	
KL coefficient / entropy coefficient	0.01 / 0.001	
Execution invalid-action cost	0.1 per invalid action	
Rollout temperature / top-p / top-k	1.0 / 1.0 / disabled	
Evaluation temperature	0.4, stochastic	
GiGPO normalization mode	mean/std	mean-only
Discount / anchor advantage weight	0.95 / 1.0	
Role weights: regular / intervention / editing	1 / 1 / 1	
Invalid edit / no-op reward	-1 / 0	
Routing: z, ρ, η	1.645, 0.25, 0.125	
Distillation turns / edit; maximum / iteration	16; 64	
Distillation temperature / mixture weight	1.2 / 0.5	
Distillation global minibatch / per-GPU microbatch	64 / 1	
Training iterations / checkpoint evaluation interval	150 / 5	

Table 3: Training configuration, shared by all backbones. Regular trajectories train acting in every method; intervention trajectories measure transfer and train acting under the edited context. SearchQA uses the WebShop settings except for 4 maximum turns and 2 external records.

Collection and infrastructure. The 128 regular trajectories are collected asynchronously in one phase, followed by the four editing requests and the 64 interventions. ALFWorld uses pinned intervention workers; WebShop allocates two of each query's eight managed workers to each edit without concurrent reuse. The supported recipe uses single-node fully sharded data parallel (FSDP) training (Zhao et al., 2023) and a standalone generation server with tensor parallelism one and data parallelism equal to the GPU count. Runtime seed, rollout iteration, store revision, edit provenance, and actor version are checkpointed.

D PROMPTS, TASK SCHEDULES, AND MEMORY UPDATES

Prompts. Prompts are copied from our implementation, with braces filled at run time; acting templates keep the wording of verl-agent's GiGPO (Feng et al., 2025).

Memory editing. The online editor (the acting policy) and the cold-start labeler receive the same system and user messages. The system message is:

```
You are the memory editor for an {environment_name} agent. Learn from eight attempts at one
     source task.
Return exactly one JSON object. Return operation=noop unless the observed trajectories
     ground one reusable memory improvement. Use operation=revise only for a memory present
     in retrieved_memories and operation=add for a new concept. Use operation=delete only to
      remove a shown memory that the evidence contradicts. Keep new_text within 128 words.
     The edit will be evaluated on other tasks, whose evidence is not available here. Do not
      infer unobserved states, actions, outcomes, or causes.
```

The user message is the compact JSON serialization of `{"group": <source dossier>, "output_schema": <schema>}`, wrapped in the actor's chat template, with no further natural-language instruction. The dossier contains the task instruction and family, the number

of successes among the eight attempts, the retrieved memories visible to those attempts (identifier, text, task families, applicability fields, and residency), and one record per attempt with its outcome, episode reward, and up to eight turns (the first and last four). Each turn records the observation, admissible actions, model response (at most 2,000 characters), executed action, format and admissibility flags, reward, and the identifiers of the visible memories. Query tasks and their outcomes are never part of the input. The dossier is fitted into 4,096 tokens by removing middle turns of the longest trajectory, keeping at least one turn per trajectory, and then shortening long strings; all eight trajectory identifiers and outcomes are always kept, and a dossier that still does not fit is rejected rather than truncated. For WebShop, hidden goal and catalog metadata are redacted. The output schema is:

```
{"operation": "noop | add | revise | delete",
 "target_memory_id": "required for revise or delete",
 "concept_key": "short snake_case concept",
 "new_text": "one reusable grounded instruction",
 "memory_type": "procedure | search | action_rule | state_tracking | failure_lesson",
 "applicability": {"description": "when it applies", "goal_terms": [], "state_cues": [], "
    anti_cues": []},
 "cited_trajectory_ids": [], "confidence": 0.0,
 "contradicts_retrieved_memory": false, "environment_specific": false,
 "rationale": "brief grounded reason"}
```

Additions and revisions require nonempty text of at most 128 words; revisions and deletions must target a memory shown in the dossier, and deletions carry no text. A no-op is `{"operation": "noop"}`.

Acting with memory. When at least one memory is retrieved, the user message begins with a memory block:

```
Relevant memory:
- {memory_text_1}
- {memory_text_2}
...

{environment_action_prompt}
```

Only memory texts are rendered, in retrieval order, with external and review records formatted identically; without retrieved memory, the message is the action prompt alone. The ALFWorld action prompt, after the first step, is:

```
You are an expert agent operating in the ALFRED Embodied Environment. Your task is to: {
    task_description}
Prior to this step, you have already taken {step_count} step(s). Below are the most recent {
    history_length} observations and the corresponding actions you took: {action_history}
You are now at step {current_step} and your current observation is: {current_observation}
Your admissible actions of the current situation are: [{admissible_actions}].

Now it's your turn to take an action.
You should first reason step-by-step about the current situation. This reasoning process
    MUST be enclosed within <think> </think> tags.
Once you've finished your reasoning, you should choose an admissible action for current step
     and present it within <action> </action> tags.
```

The WebShop action prompt, after the first step, is:

```
You are an expert autonomous agent operating in the WebShop e-commerce environment.
Your task is to: {task_description}.
Prior to this step, you have already taken {step_count} step(s). Below are the most recent {
    history_length} observations and the corresponding actions you took: {action_history}
You are now at step {current_step} and your current observation is: {current_observation}.
Your admissible actions of the current situation are:
[
{available_actions}
].

Now it's your turn to take one action for the current step.
You should first reason step-by-step about the current situation, then think carefully which
     admissible action best advances the shopping goal. This reasoning process MUST be
    enclosed within <think> </think> tags.
Once you've finished your reasoning, you should choose an admissible action for current step
     and present it within <action> </action> tags.
```

The history window is two steps, each rendered as `[Observation {i}: '{observation}', Action {i}: '{action}']`. At the first step, the sentences about prior steps are omitted and the current observation, which contains the task, is given directly; a WebShop prompt longer than 13,000 characters falls back to this form. The Search-QA action prompt, after the first step, is:

```
You are an expert agent tasked with answering the given question step-by-step.
Your question: {task_description}

Prior to this step, you have already taken {step_count} step(s). Below is the interaction
     history where <search> </search> wrapped your past search queries and <information> </
     information> wrapped the corresponding search results returned by the external search
     engine. History:
{search_history}

Now it's your turn to respond for the current step.
You should first conduct reasoning process. This process MUST be enclosed within <think> </
     think> tags.
After completing your reasoning, choose only one of the following actions (do not perform
     both):
(1) If you find you lack some knowledge, you can call a search engine to get more external
     information using format: <search> your query </search>.
(2) If you have enough knowledge to answer the question confidently, provide your final
     answer within <answer> </answer> tags, without detailed illustrations. For example, <
     answer>Beijing</answer>.
```

Each history entry is rendered as `[Search {i}: '{search query}', Information {i}: '{search results}']`, the history paragraph is omitted at the first step, and the retrieved-memory block is prepended as above. There is no admissible-action list: a response is a `<think>` block followed by exactly one `<search>` or `<answer>`; the gold answer is used only for scoring.

Cold-start labeler. The labeler (DeepSeek-V4-Flash (DeepSeek-AI et al., 2026)) is sampled on the editor messages above at temperature 0 in JSON mode. A response is accepted only if it parses as a single JSON object, gives a nonempty rationale and a confidence in $[0, 1]$, cites source trajectories explicitly for every edit other than a no-op, provides the applicability description and cue lists for additions and revisions, and fits in 512 actor tokens. Invalid or truncated responses are rejected, not repaired. Online parsing is more lenient: it also accepts a fenced JSON object and treats missing citations as citing all eight source trajectories.

Consolidation. Distillation reuses the action prompt unchanged; on a cached query turn, the teacher reads the student's context with the candidate overlaid:

```
Student: {cached_memory_block}

{cached_environment_action_prompt}

Teacher: {memory_block_with_candidate}

{same_cached_environment_action_prompt}
```

An addition appends the candidate to the teacher's memory block and a revision replaces its parent there; the student keeps the original block, which may still contain the parent. Both branches score the same cached response tokens.

Task manifests and sampling schedule. Each environment's training split is partitioned once into exclusion groups (Appendix E) and stored as a manifest of group and task identifiers. Each seed fixes a random order over groups; an iteration takes the next 16, draws one task from each, and splits them eight/eight into source and query folds by a seeded permutation. When the order is exhausted, a new epoch begins with a fresh permutation, so a task can be a query in one iteration and a source in a later one, but never both within an iteration. All methods trained in our pipeline consume the same per-seed task sequence.

Conflicts between accepted edits. Two accepted edits in the same iteration conflict when they revise or delete the same parent record, or when a new record shares a concept identifier with another accepted record in the same pool. Only the edit with the larger lower bound L_e is committed (ties broken by $\widehat{U}(e)$, then by stable edit identifier). A deletion never overrides a revision of the same

parent that is routed to review. Edits that lose a conflict keep their editing reward, so every returned edit response contributes to learning (Algorithm 1, line 12).

Release. We will release the code, the manifests and per-seed task sequences, and the final memory stores of all reported runs.

E EVALUATION PROTOCOL

Data separation. Training, development, diagnostic, and final evaluation tasks come from separate manifests; source and query folds are both training data. To prevent leakage between folds, ALFWorld tasks sharing a FloorPlan are assigned to the same fold; WebShop tasks are grouped by product identity, or by a canonical goal signature when the product is unavailable; Search-QA questions are grouped by the normalized answer string and the Wikipedia entity linked to the question. Final ALFWorld results use the seen evaluation split (`valid_seen`), disjoint from all training tasks. Thresholds and budgets were chosen on development tasks, and each final configuration was evaluated once.

Seeds and checkpoints. Training seeds 0, 1, and 2 control task order, the source/query partition, and rollout sampling. All methods start from an empty memory store, except GiGPO + static memory, whose store is written once before training. We report the final checkpoint (iteration 150) for all methods. The actor and store are never modified during evaluation.

Controlled baselines. All methods in Table 2 share MECA's base model, prompt format, context limits, rollout sampling, evaluation schedule, and ordinary update settings (Table 3): 16 tasks $\times$ 8 rollouts (128 regular trajectories) per iteration for 150 iterations, except for the budget-matched controls below. MECA additionally collects 64 intervention rollouts per iteration. GRPO and GiGPO use no memory. *GiGPO + labeler SFT* fine-tunes on the 2,048 cold-start demonstrations for two epochs (learning rate 10^{-5}, batch size 64) before GiGPO. *GiGPO + static memory*: the cold-start labeler writes edits from source trajectories of the initial policy on 512 training tasks; valid edits form a fixed external store retrieved with the same retriever and budget as MECA. *GiGPO + unfiltered memory*: the shared actor proposes edits from the same dossiers and with the same cold start as MECA; every valid, non-conflicting edit is committed to external memory, the editing role receives no reward, and no intervention rollouts are collected. *ExpeL* (Zhao et al., 2024) (Table 5) extracts insights from base-policy trajectories on the same training tasks and retrieves them with MECA's retriever and budget.

Budget-matched controls (Figure 4 and Table 12). *GiGPO, 12 rollouts* and *GiGPO, 225 iterations* give GiGPO as many environment trajectories as MECA: the first samples 12 rollouts for each of the 16 tasks per iteration (192 trajectories) for 150 iterations, and the second keeps 16 tasks $\times$ 8 rollouts and trains for 225 iterations. *No transfer credit* keeps every component of MECA, including intervention rollouts and their loss, the editing loss, and the distillation budget, but assigns each valid edit the transfer reward of another valid edit from the same iteration, drawn by a seeded permutation, and assigns routes at random with the full method's per-environment route shares; invalid edits and no-ops keep their fixed rewards. *No consolidation, step-matched* routes review edits to external memory and replaces each distillation step with an additional ordinary step on the same batch. Random selection and placebo memory are defined in Table 13.

Metrics. ALFWorld success is computed over 128 `valid_seen` episodes per seed. WebShop reports the mean normalized task score and exact success separately. Search-QA reports exact-match accuracy per dataset and their average weighted by evaluation-set size: $\mathrm{Avg} = \sum_d n_d \, \mathrm{EM}_d / \sum_d n_d$. The weights n_d are 3,610 (NQ), 11,313 (TriviaQA), 14,267 (PopQA), 7,405 (HotpotQA), 12,576 (2Wiki), 2,417 (MuSiQue), and 125 (Bamboogle), totaling 51,713 questions. Table 1 recomputes every method's average, including published ones, from its displayed per-dataset scores, rounded to one decimal place. All differences are in percentage points.

Uncertainty. Methods share seeds and evaluation tasks, so differences are paired. Confidence intervals for differences use a two-level bootstrap with 2,000 resamples: training seeds are resampled with replacement, and within each resampled seed, evaluation tasks are resampled in clusters of the same exclusion identity, preserving the pairing between methods and the shared controls. Statistics over edits (correlations and route means) resample runs and then edits within runs. Primary comparisons

were fixed before running them; other intervals are reported without multiple-comparison correction and should be read as descriptive.

F ADDITIONAL RESULTS

Controlled comparison. Table 4 evaluates the memory baselines with their memory withheld, and Table 5 reports training-free memory.

	Qwen2.5-7B			Qwen2.5-3B			Qwen3-1.7B		
Memory removed at evaluation	ALFW	WebS	SQA	ALFW	WebS	SQA	ALFW	WebS	SQA
GiGPO (no memory)	87.5	72.7	47.2	85.9	69.5	43.7	72.7	46.9	41.9
GiGPO + static memory	90.4	74.2	46.8	85.9	69.8	42.7	79.9	46.6	41.1
GiGPO + unfiltered memory	88.8	72.9	46.1	83.9	67.4	42.1	78.6	51.0	40.9
MECA	95.8	84.1	49.6	92.7	75.0	46.4	89.8	78.9	43.9

Table 4: **Memory methods evaluated with all memory withheld** (%).

	Qwen2.5-7B			Qwen2.5-3B			Qwen3-1.7B		
Method	ALFW	WebS	SQA	ALFW	WebS	SQA	ALFW	WebS	SQA
GiGPO	87.5	72.7	47.2	85.9	69.5	43.7	72.7	46.9	41.9
ExpeL (no training)	46.9	27.3	36.8	21.9	12.5	30.4	12.5	7.8	27.9

Table 5: **Training-free memory** (%). ExpeL (Zhao et al., 2024) builds memory from base-policy trajectories on the training tasks without updating the policy.

Cost and sample efficiency. Edit generation, intervention rollouts, and distillation raise wall-clock time per iteration from 183 to 245 seconds (Figure 6b). At equal environment trajectories (Figure 6a), MECA leads the stronger control by 7.3 points on ALFWorld; 12 rollouts per task and 225 iterations close only 7% and 15% of the GiGPO–MECA gap. MECA reaches GiGPO's final ALFWorld success with 50% of its trajectories, although a third of them measure transfer.

Method	Env. actions (M)	Iterations
GRPO	0.44	150
GiGPO	0.41	150
GiGPO, 12 rollouts	0.62	150
GiGPO, 225 iterations	0.60	225
GiGPO + static memory	0.40	150
GiGPO + unfiltered memory	0.39	150
MECA	0.55	150

Table 6: Training cost (Qwen2.5-3B-Instruct, ALFWorld). Baselines collect 16 tasks $\times$ 8 rollouts per iteration; MECA adds 64 intervention rollouts; the GiGPO controls use 12 rollouts per task or 225 iterations. The 2,048 labeler calls for cold-start demonstrations are excluded.

G TRANSFER AND CONSOLIDATION DIAGNOSTICS

These protocols and results underlie Section 2 and Section 4.4.

Candidate sampling. For Section 2, candidates are additions written by DeepSeek-V4-Flash from base-policy trajectories and evaluated with the base policy; for Section 4.4, they are edits written by MECA at iterations 50, 100, and 150. Sampling, fixed before any probe outcome is observed, is stratified by operation, source task family, and routing outcome, and results are reweighted by sampling probability. Consolidation diagnostics use additions and revisions only, since deletions are not distilled.

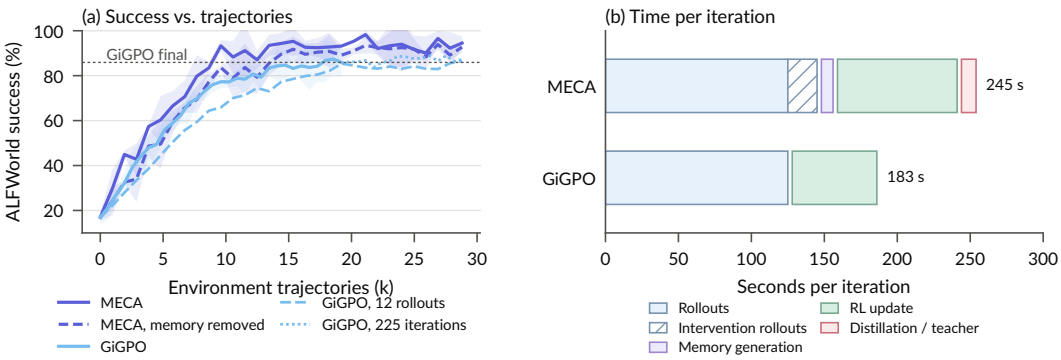


Figure 6: **Sample efficiency and cost** (Qwen2.5-3B-Instruct, ALFWorld). (a) Success against environment trajectories; the trajectory-matched GiGPO controls use 12 rollouts per task or 225 iterations. (b) Wall-clock time per iteration by stage. Shading: std over seeds.

Source benefit versus transfer (Figure 2a). *Pools.* Within each environment's training split, each exclusion group (FloorPlan or product) is assigned to one pool, source (50%), query (25%), or probe (25%), stratified by task family (six ALFWorld task types; five WebShop product categories); official evaluation splits are never used. *Source tasks.* A screening pass (8 rollouts, discarded afterwards) keeps source tasks with base success between 0.1 and 0.9, avoiding ceiling and floor effects; we sample 210 ALFWorld and 200 WebShop source tasks, one edit each. *Edits.* Each addition is written from 8 base-policy rollouts on its source task; no-ops are excluded, and the base store is empty in all conditions. *Source benefit* uses 32 fresh rollouts with and 32 without the edit (paired environment seeds), never the rollouts that produced the edit; a second independent batch gives test–retest reliability. *Transfer* uses 12 probe tasks from the edit's task family with 8 rollouts per condition; split-half reliability is computed over two halves of 6 probe tasks. *Selection.* Source- and query-based selection retain the same fraction of edits and are evaluated on the same probe tasks. The query panel has 8 same-family query tasks with 2 rollouts per condition; a separate arm scores each edit on a single query task with 32 rollouts per condition, matching the source panel, to separate disjointness from breadth. Confidence intervals use 2,000 bootstrap resamples over edits. Cross-family probes (not shown) give transfer near zero regardless of source benefit.

Calibration of online transfer (Figure 7a). We sample 100 proposals per seed from MECA runs, stratified by route, and compare the online estimate $\widehat{U}(e)$, computed from 8 queries with 2 rollouts each, against read value on an independent probe panel with 32 rollouts per condition. We report the Spearman correlation with a two-level bootstrap CI (runs, then proposals) and the per-edit standard error of $\widehat{U}(e)$.

Fresh-panel re-estimation (Table 7). After routing, each calibration proposal is re-scored with the online design on 8 fresh query tasks from the diagnostic manifest (2 intervention rollouts each, new base-context controls) at the same checkpoint and store. The online-minus-fresh difference, averaged within a route, measures selection optimism; we also compare it descriptively across training thirds. Three end-to-end variants decouple reward from selection: splitting the 8 queries into disjoint halves for the editing reward and for routing (same budget), adding an independent panel of 8 queries used only for routing (320 trajectories per iteration), and replacing shared controls with two candidate-specific control rollouts per query (256 trajectories per iteration).

Reliability of the transfer estimate. The online estimate $\widehat{U}(e)$ correlates with independent-probe read value at Spearman 0.68 (95% CI [0.59, 0.75]; Figure 7a). Because the same outcomes train the editor and select edits, the estimates of selected edits are optimistic: on a fresh panel, review-routed edits shrink from 36.7 to 22.8 points, yet 96% remain positive and they still transfer far more than external edits (9.5 points; Table 7). The shrinkage is U-shaped over training (16.4, 9.3, and 16.0 points in the three thirds), but with only 8 review-routed edits per third this neither establishes a trend nor rules out increasing exploitation of panel noise. Decoupling reward from selection changes end-to-end success little (Qwen2.5-3B-Instruct, ALFWorld and WebShop): scoring reward and routing on disjoint halves of the panel costs 0.7 points with memory, as expected from halving each

panel, while routing on an independent panel of 8 further queries or giving each candidate its own controls changes success by at most 0.5 points with or without memory.

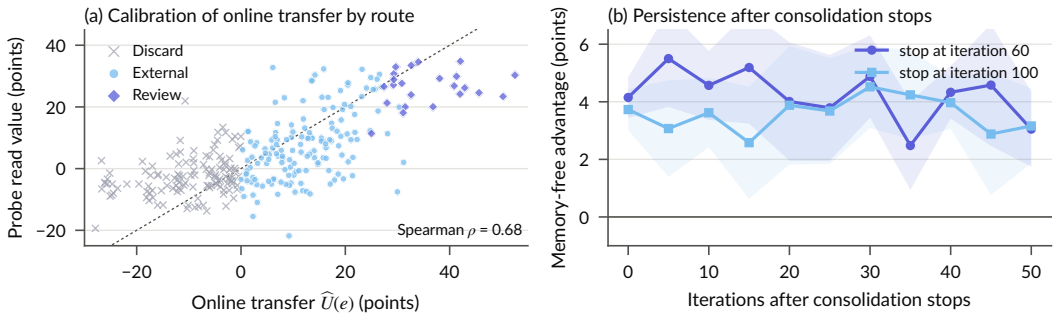


Figure 7: **Calibration and persistence** (Qwen2.5-3B-Instruct). (a) Calibration proposals on ALFWorld and WebShop; read value is measured on an independent probe panel. (b) Advantage of the consolidated branch over matched ordinary updates on a frozen bank of 64 memories. Shading: std over seeds.

Route	n	Online $\widehat{U}(e)$	Fresh panel	Independent probe	Online − fresh
Review	24	36.7	22.8	26.3	+13.9
External	151	12.3	9.5	7.4	+2.8
Discarded	125	−10.2	−1.3	−2.5	−8.9
Review, iterations 1–50	8	38.2	21.8	26.8	+16.4
Review, iterations 51–100	8	33.9	24.6	28.4	+9.3
Review, iterations 101–150	8	37.9	21.9	23.7	+16.0

Table 7: **Selection bias of the online transfer estimate** (points; the 300 calibration proposals of Figure 7a). *Online*: the reward and routing estimate (8 queries × 2 rollouts, shared controls). *Fresh panel*: the same design on 8 new queries with new controls, drawn after routing. *Independent probe*: read value on 12 tasks, 32 rollouts per condition. The last column is the optimism from selecting on the online estimate.

Learning value (Figure 2b and Table 8). For each candidate, every branch is cloned from the same actor and optimizer state (the base policy for Section 2; post-RL MECA checkpoints for Section 4.4). For Figure 2b, the pool contains 120 additions drawn uniformly from the Observation 1 edits (half per environment); each is consolidated in its own branch, and learning value is measured on two independent probe panels of 12 same-family tasks with 8 rollouts per branch. Every rule selects the top 25% of the pool: by transfer on the query panel (the estimate available during training), uniformly at random (reported as the pool mean), by source benefit, or by learning value on the first panel (oracle); all rules are scored on the second panel, so the oracle is not favored by selecting on noise. Confidence intervals use 2,000 bootstrap resamples of the pool. As these comparisons are between distillation branches, they do not depend on the ordinary-update control. The *matched ordinary update* performs an RL update on the same cached turns with the same number of gradient steps and training tokens as distillation and serves as the reference for G_t. The *fresh-rollout RL* branch instead collects new on-policy rollouts on the same tasks and performs ordinary GiGPO updates until it matches the distillation branch's wall-clock time; it is the stronger compute-matched control. Probe tasks for Table 8 are screened at each checkpoint like source tasks (base success in $[0.1, 0.9]$). *Off-family* probes are 12 tasks from other task families, evaluated without the candidate, to detect interference. The *placebo* and *random* branches distill, respectively, a length-matched memory from an unrelated task family and randomly chosen valid additions and revisions, with the same budget and cached contexts. All branches are evaluated on the same independent probe panel with and without the candidate, keeping other memories fixed. With the four success means $J_{0,-}, J_{0,+}, J_{e,-}, J_{e,+}$ (control/candidate branch, candidate absent/present),

$$G = J_{e,-} - J_{0,-}, \qquad D_0 - D_e = G - (J_{e,+} - J_{0,+}), \tag{15}$$

where $D_b = J_{b,+} - J_{b,-}$. The identity shows that a smaller gap D_e alone is ambiguous: it can arise from lower candidate-present success. We therefore report absolute success in both conditions alongside G_t and D_t^{post}.

Consolidation results (Table 8). Relative to the matched ordinary update, selected-candidate distillation lowers the remaining benefit of reading the candidate, $D_t^{\text{post}}(e) = \mathbb{E}_q[V_{\theta^e}(q, C_q \oplus e) - V_{\theta^e}(q, C_q)]$, from 9.2 to 6.7 points, through higher candidate-absent success (56.3 vs. 53.1) rather than lower candidate-present success (63.0 vs. 62.3). Learning value is largest at iteration 50 and remains positive at iteration 150 (2.2). Masking paraphrases and related memories (below) changes this by less than 0.6 points, so they do not explain the effect. On probes from other task families, selected-candidate distillation changes success by -0.3 points (95% CI $[-1.1, +0.5]$), compared with -0.2 for the matched ordinary update.

	Pooled over checkpoints			G_t **by checkpoint**			
Update from a common checkpoint	$-e$	$+e$	D_t^{post}	50	100	150	Off-family
No additional update	52.4	61.8	9.4	–	–	–	–
Matched ordinary update (cached turns)	53.1	62.3	9.2	0 (ref.)	0 (ref.)	0 (ref.)	-0.2
Fresh-rollout RL, compute-matched	53.8	62.7	8.9	0.9	0.7	0.5	$+0.1$
Placebo memory distillation	53.3	62.1	8.8	0.3	0.2	0.1	-0.4
Random candidate distillation	53.9	62.5	8.6	1.0	0.8	0.5	-0.6
Selected candidate distillation	**56.3**	63.0	6.7	**4.1**	**3.3**	**2.2**	-0.3

Table 8: **Learning value and memory dependence** (Qwen2.5-3B-Instruct; ALFWorld and WebShop candidates). $-e$/$+e$: probe success (%) without/with the candidate; G_t (Eq. 2): relative to the matched ordinary update; D_t^{post}: remaining benefit of reading the candidate; Off-family: success change (points) on other task families, relative to no additional update.

Concept masking. So that related memories cannot substitute for a removed candidate, memory-removed evaluation also masks its paraphrases (BGE-M3 (Chen et al., 2024) similarity above 0.85), its parent and child records, and memories sharing its concept identifier; on 400 hand-labeled pairs, the matcher has 91% precision and 84% recall.

Persistence (Figure 7b). A bank of 64 candidate memories, base contexts, and probe tasks is frozen. Two branches are cloned from a common actor and optimizer state; one adds consolidation of the bank to ordinary training, and the other adds matched ordinary updates. Consolidation stops at one of two iterations fixed in advance (60 and 100), after which both branches continue ordinary training for 50 iterations and are evaluated every 5 iterations with and without the bank; each setting is repeated over the training seeds.

Editing quality (Figure 5a and Table 9). Writer checkpoints (iterations 0, 25, 50, 75, 100, 150) of MECA and of the source-reward variant produce edits for a fixed set of 200 source dossiers not seen in training. Each edit's transfer is measured on independent probes with a fixed base store and one of three frozen readers: the base policy, which favors neither writer, the final source-reward policy, and the final MECA policy. We report transfer per allocated edit slot. Table 10 gives the per-checkpoint results under the base-policy reader. Edits from the transfer-reward writer transfer better under every reader, including the reader trained with source-task reward (6.9 vs. 1.9 points), and the final MECA writer surpasses the cold-start labeler by 3.3 points under the base-policy reader. Readers also improve: the same edits transfer 2.1 points more when read by the final MECA policy, consistent with the intervention loss teaching the policy to use new memories.

Edit annotation and audit (Figure 5b). We sample 50 accepted edits per checkpoint, stratified by route. Two annotators, blind to route, checkpoint, and variant, label each edit as *episode-specific* (refers to a particular layout, object instance, or product), *conditional* (a procedure with an explicit applicability condition), or *general*; disagreements are resolved by discussion, and an LLM judge (DeepSeek-V4-Flash) with the same rubric labels the remaining edits after validation against the annotators (Cohen's κ between 0.71 and 0.78 across environments (Cohen, 1960); Table 11). The audit additionally flags generic filler (guidance without task content) and environment shortcuts (e.g., always selecting the first search result). At iteration 150, pooling the 50 audited edits per environment gives 7/150 generic-filler edits (4.7%) and 2/150 environment shortcuts (1.3%; Table 11); these are

	Reader		
Writer	Base policy	Source-reward final	MECA final
Cold-start labeler (DeepSeek-V4-Flash)	3.1	3.6	4.4
Source-reward variant, final	1.2	1.9	2.3
MECA, final	**6.4**	**6.9**	**8.5**

Table 9: **Writer–reader transfer** (points; Qwen2.5-3B-Instruct, mean over ALFWorld and WebShop). Each cell is the mean transfer per edit slot of edits written by the row's writer and read by the column's policy.

Writer checkpoint	0	25	50	75	100	150
MECA (transfer reward)	2.7	3.5	4.4	5.3	5.9	6.4
Source-task reward	2.7	2.2	1.8	1.5	1.3	1.2

Table 10: **Transfer per edit slot by writer checkpoint** (points; base-policy reader; Qwen2.5-3B-Instruct, ALFWorld and WebShop).

unweighted pooled counts. Mean edit length changes by only 5 words over training, so transfer reward does not simply favor longer text.

Category (% of accepted edits)	ALFWorld	WebShop	Search-QA
Episode-specific / conditional / general	14 / 46 / 40	20 / 48 / 32	22 / 52 / 26
Generic filler	4.0	4.0	6.0
Environment shortcut	2.0	2.0	0.0
LLM judge vs. annotators (κ)	0.78	0.74	0.71

Table 11: **Edit audit** of MECA's accepted edits at iteration 150 (Qwen2.5-3B-Instruct; 50 edits per environment). Percentages are unweighted counts out of 50 (2% per edit); the text pools all 150 edits.

H Ablation definitions and additional analyses

Table 12 gives the final values behind Figure 4, with and without memory, plus variants not plotted there; Table 13 defines them. All MECA variants keep the full method's 128 regular trajectories, 64 intervention rollouts, update schedule, and seeds, so differences are paired; Table 14 replicates the three primary comparisons on Qwen3-1.7B.

Further variants. Removing the editing loss lowers memory-free success by 3.1 points on average, and the point estimate is negative on every benchmark with and without memory, suggesting that shared parameters alone do not teach editing. Removing the intervention loss lowers memory-free success by 1.8 points (most on WebShop), and allowing only additions lowers success by 0.5–2.6 points; these estimates suggest, but do not establish, benefits from the intervention loss and from revisions and deletions. Without cold-start initialization, MECA still improves over GiGPO by 3.3 points on average without memory. Keeping consolidated memories external changes success by at most 1.3 points (slightly higher with memory, slightly lower without) and uses 38% more context tokens. Training a separate editor of the same size lowers success by 0.4 points on average with memory while doubling the trainable parameters.

Routing thresholds. Figure 8 varies the review threshold η and the confidence multiplier z one at a time. WebShop success changes by at most 2.3 points (at $z = 0$) and Search-QA exact match by at most 0.9.

Distillation data. Distilling on cached base-context turns and on successful memory-conditioned trajectories gives similar learning value (3.2 vs. 3.0 points) under equal token and update budgets, while the cached turns require no additional sampling.

Variant	With memory			Memory removed		
	ALFW	WebS	SQA	ALFW	WebS	SQA
Full MECA (absolute, %)	94.5	77.3	47.6	92.7	75.0	46.4
Memory-free RL with matched trajectories						
GiGPO	–	–	–	−6.8	−5.5	−2.7
GiGPO, 12 rollouts	–	–	–	−6.2	−4.4	−3.6
GiGPO, 225 iterations	–	–	–	−5.5	−3.6	−3.3
Editing signal						
No transfer credit	−4.7	−4.1	−3.1	−4.9	−4.2	−3.0
Source-task reward*	−5.5	−6.5	−2.5	−6.0	−7.3	−2.9
Evidence-overlapping queries	−3.6	−4.2	−1.7	−4.2	−4.9	−2.0
Source benefit for reward and routing	−6.5	−7.8	−3.1	−8.3	−9.1	−3.6
Consolidation						
No consolidation*	−1.3	−1.8	−0.7	−5.5	−6.5	−2.6
No consolidation, step-matched	−1.3	−1.5	−0.6	−5.2	−6.2	−2.5
Distill all accepted edits	−0.8	−1.3	−0.4	−3.1	−3.9	−1.5
Random selection*	−1.0	−1.6	−0.5	−4.7	−5.5	−2.1
Placebo memory	−1.3	−1.8	−0.6	−5.2	−6.2	−2.4
Memory operations						
Additions only	−1.8	−2.6	−0.6	−1.6	−2.3	−0.5
No external pool	−2.1	−2.6	−1.0	−0.5	−0.5	−0.2
Other components						
No editing loss	−3.1	−3.9	−1.4	−3.4	−4.4	−1.6
No intervention loss	−2.3	−2.9	−1.1	−1.8	−2.6	−0.9
No cold start	−1.6	−2.1	−0.8	−1.8	−2.3	−0.9
Consolidate, keep external	+0.3	+0.3	+0.1	−0.8	−1.3	−0.4
Separate editor	−0.5	−0.5	−0.3	−0.5	−0.8	−0.2

Table 12: **Ablations and budget-matched controls: final values** (Qwen2.5-3B-Instruct; curves for all but the last group in Figure 4). The first row gives MECA's success (%); other rows give the change from it (points). *Primary comparisons, fixed before the runs. Definitions are in Appendix E and Table 13.

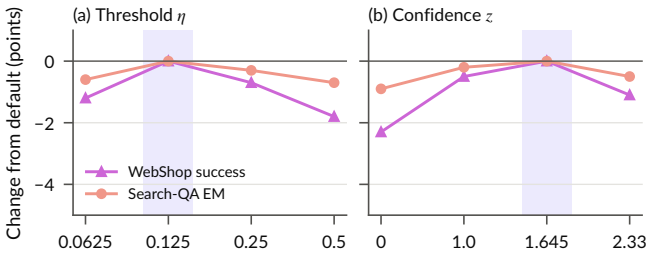


Figure 8: **Sensitivity** (Qwen2.5-3B-Instruct; shaded: default setting). One setting varies at a time.

Variant	Intervention	What it tests
Source-task reward	Replace the editing reward with source benefit, estimated on fresh replays of the edit's source task with as many rollouts as the transfer estimate; admission and routing still use transfer measured on query tasks.	Transfer vs. source benefit as the editing signal.
No editing loss	$\lambda_{\text{edit}} = 0$; edits are still proposed, scored, routed, and consolidated.	Direct editing credit beyond shared parameters.
No intervention loss	$\lambda_{\text{aug}} = 0$; intervention rollouts are still used for rewards and routing.	Practicing with new memories.
No cold start	Skip supervised initialization of editing.	Dependence on labeler demonstrations.
Evidence-overlapping queries	Score each edit on query tasks that include the dossier's source tasks, with the same rollout budget.	Disjointness vs. scoring on other tasks.
Source benefit for reward and routing	Use source-task benefit for both the editing reward and routing.	Transfer as the selection signal.
Additions only	Revisions and deletions are disallowed in the editing prompt and rejected.	Value of revising and deleting.
No external pool	Edits that would be routed external are discarded.	Deployment value of uncertain edits.
No consolidation	Route all accepted additions and revisions to external memory; no distillation.	Value of consolidation.
Random selection	Distill uniformly sampled valid additions/revisions with the same total budget.	Selection quality, at equal compute.
Placebo memory	Distill with a length-matched memory from an unrelated task family in the teacher context.	Content vs. distillation as regularization.
Distill all accepted	Distill every accepted addition and revision (budget grows accordingly).	Need for selection when budget is not binding.
Consolidate, keep external	Same distillation as the full method, but selected memories stay in external memory.	Effect of removing memories from context.
Separate editor	A second model of the same size proposes edits and is trained with the editing loss; the actor keeps the regular and intervention losses.	Cost or benefit of parameter sharing.

Table 13: Ablation definitions (results in Table 12). Budget-matched controls, no transfer credit, and step-matched no consolidation are defined in Appendix E.

	With memory			Memory removed		
Variant	ALFW	WebS	SQA	ALFW	WebS	SQA
Full MECA (absolute)	91.4	81.3	44.8	89.8	78.9	43.9
Source-task reward	−6.2	−7.8	−2.2	−7.0	−8.3	−2.6
No consolidation	−1.6	−2.3	−0.6	−6.2	−7.6	−2.3
Random selection, matched budget	−1.3	−1.8	−0.5	−4.9	−6.0	−1.9

Table 14: **Primary ablations on Qwen3-1.7B** (first row absolute %, other rows change in points).

I RETRIEVER ROBUSTNESS

The dense retriever embeds the task prompt and each record's text and applicability fields with BGE-M3 (Chen et al., 2024) and ranks eligible records by cosine similarity, with the same environment, status, and anti-cue filters, the same task-family bonus, and the same budgets as the lexical retriever. Training and evaluating with it changes MECA's success by +0.4 points with memory and +0.3 without; swapping it in only at evaluation, on the final store of a lexical-retriever run, lowers success with memory by 0.5 points (Table 15).

		With memory		Memory removed	
Method	Retriever	ALFW	WebS	ALFW	WebS
GiGPO + unfiltered memory	Lexical (default)	89.1	73.4	83.9	67.4
GiGPO + unfiltered memory	Dense, train and evaluate	89.6	74.2	84.1	67.7
MECA	Lexical (default)	94.5	77.3	92.7	75.0
MECA	Dense, train and evaluate	94.8	77.9	93.0	75.3
MECA	Dense, evaluation only	94.0	76.8	–	–

Table 15: **Retriever robustness** (Qwen2.5-3B-Instruct, %). *Dense*: BGE-M3 cosine similarity with the lexical retriever's filters and budgets; *evaluation only*: swapped in for the final store without retraining.

J LIMITATIONS AND EXAMPLES

Limitations. Transfer is estimated from a finite query panel, so the estimates carry sampling noise and may under-represent task families outside the panel; the uncertainty adjustment is a simple heuristic, and a more principled treatment of selection effects is a natural extension (Appendix G). Routing is decided once per memory, while a memory's value may shift as the policy and the rest of the store evolve; revisiting earlier decisions is a promising direction. Distillation uses cached prefixes and the policy itself, reading the memory, as the teacher, so its quality depends on how well the policy interprets each memory. Review-routed memories leave the deployment context at routing time; verifying consolidation before removal would make this step more conservative. Finally, each run trains on a single environment; continual and multi-domain training are left to future work.

Examples. Table 16 shows actor-written records illustrating action preconditions and an object-type distinction on ALFWorld and a required product option on WebShop.

Record	**Stored text (verbatim)**
ALFWorld, clean task action rule external pool	To clean a mug, you must be holding the mug (e.g., after 'take mug 1 from countertop 1') and be at the sinkbasin; then use 'clean mug 1 with sinkbasin 1'. Do not attempt to clean a mug that is not in your inventory.
ALFWorld, pick-and-place task action rule review pool	To complete a task requiring toiletpaper, you must pick up an object named 'toiletpaper' (e.g., 'toiletpaper 1'), not a 'papertowelroll'. A papertowelroll is a different object and cannot be placed on a toiletpaperhanger. When searching, check shelves and other receptacles for an object explicitly named 'toiletpaper'.
WebShop procedure external pool	When a product page shows size options (e.g., 40x40x40cm, 110x40x40cm), click the exact size option that matches the task requirement before clicking 'Buy Now'.

Table 16: **Example memory records** (Qwen2.5-3B-Instruct actor), verbatim from MECA memory stores. The pool is the recorded routing decision; it does not by itself show that a review record was consolidated.