MEMORBIT: TRACEABLE DUAL-LOOP MEMORY REINFORCEMENT LEARNING FOR LIFELONG AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Agents operating across a sequence of tasks must learn both how to use past experience and which experiences to retain. These decisions are coupled: retained memories shape the evidence available for future tasks, while an agent's memory-use behavior shapes the feedback available for updating those memories. We propose *MemOrbit*[1], a traceable dual-loop reinforcement learning (RL) framework for jointly optimizing memory-management behavior and persistent memory. The policy loop uses RL to learn memory-access and management decisions expressed as tool calls. The memory loop uses RL to update entry utilities from citation-attributed task feedback, guiding retention and memory updates. The loops share task outcomes but assign credit separately to policy decisions and persistent entries. Within each training step, the agent searches and reads a fixed, versioned memory snapshot, with provenance-bound citations linking its answer to the entries read. Persistent-memory changes are committed only at step boundaries, making successive policy–memory states auditable. Experiments on MemoryBench show that *MemOrbit* improves performance over existing memory-augmented agent approaches, demonstrating that explicit credit assignment over both agent behavior and persistent memory enables more adaptive and reliable long-horizon learning.

1 INTRODUCTION

Agents that interact with environments over long horizons can accumulate experience in external memory and reuse it across tasks. However, larger memory does not always lead to better performance. Redundant, outdated, or irrelevant entries can obscure useful information and degrade decisions. Effective memory-augmented agents must therefore learn both how to use existing memories and which experiences to retain. These two objectives are coupled: the memory bank determines the available evidence, while the policy determines which memories are retrieved, referenced, and updated. This coupling makes credit assignment challenging. A task-level reward does not reveal which actions or memory entries contributed to the outcome, and uniform credit assignment can reinforce inefficient interactions or suboptimal memory updates.

Recent work studies different aspects of memory-augmented agents. MemRL learns episodic memory utilities from interaction feedback while keeping model parameters fixed (Zhang et al., 2026c). NapMem learns tool-based navigation over structured memory (Xu et al., 2026). ECHO studies policy credit assignment across multi-turn interactions under limited context (Xie et al., 2026). However, these approaches focus on memory usage or memory learning separately. Efficient retrieval does not guarantee useful memory contents, and high-quality memories do not guarantee effective utilization. We propose **MemOrbit**, a dual-loop reinforcement learning framework that jointly learns memory usage and memory evolution. At training step t, the policy π_{θ_t} interacts with a fixed memory snapshot B_t. The *policy*

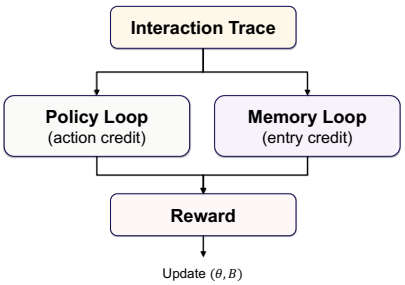


Figure 1: **Overview of MemOrbit.** The Policy loop learns action-level decisions. The memory loop learns entry-level updates from interaction traces and rewards.

[1]The codes will be released if the paper is accepted.

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.

loop optimizes decisions for searching, reading, and proposing memory operations. The *memory loop* updates memory utilities based on task feedback associated with cited entries, guiding retention and memory modification. The two loops share interaction traces while assigning credit to different units: policy actions and persistent memories.

MemOrbit uses a structured interaction protocol for memory updates. Citations are derived from successful reads in the current memory snapshot, and memory operations are committed only at step boundaries. This produces a sequence of consistent policy-memory states (θ_t, B_t) and supports analysis of how decisions and memory changes affect performance. After training, the policy can be fixed while the memory loop continues to adapt the memory bank.

Our contributions are: (1) we formulate memory-augmented agent learning as a joint optimization problem over policy behavior and memory evolution; (2) we introduce MemOrbit, a dual-loop RL framework with separate credit assignment for actions and memory entries; (3) we develop a structured memory update mechanism that enables controlled learning of persistent memory.

Related Work

External memory and experience organization. External memory has been widely used to augment language models with reusable knowledge and experience. Retrieval-augmented generation extends language models with external knowledge sources (Lewis et al., 2020), while agent-memory systems organize persistent observations, reflections, skills, and insights (Park et al., 2023; Packer et al., 2024; Shinn et al., 2023; Wang et al., 2024; Zhao et al., 2024). Recent work explores memory evolution through forgetting-based retention, linked-note organization, conversational consolidation, and temporal knowledge structures (Zhong et al., 2024; Xu et al., 2025; Chhikara et al., 2025; Rasmussen et al., 2025). Other approaches curate reusable strategies from trajectories or evolve memory structures themselves (Zhang et al., 2026b; Ouyang et al., 2026; Zhang et al., 2026a). These methods mainly focus on memory representation and organization, whereas MemOrbit studies how a fixed memory interface can be jointly optimized with policy learning and persistent entry updates.

Learning memory management and utilities. Recent work applies reinforcement learning and outcome-driven optimization to memory operations. Memory-R1 and Mem-α learn memory construction or management from downstream task rewards (Yan et al., 2026; Wang et al., 2025), while MemBuilder and Fine-Mem study intermediate rewards and operation-level attribution (Shen et al., 2026; Ma et al., 2026a). Other methods learn external memory adaptation from successful and failed experiences (Zhang et al., 2023; Zhou et al., 2025; Wang, 2026). MemRL and MemQ further learn memory-entry utilities and propagate credit through memory dependencies (Zhang et al., 2026c; Liao et al., 2026). Unlike these approaches, MemOrbit separates credit assignment for policy actions and persistent memory entries: the policy loop learns memory interaction decisions, while the memory loop updates entry utilities after memory changes are committed.

Tool use, traceable credit, and evaluation. RL-based tool use and memory navigation improve agents' ability to actively retrieve and manipulate information (Jin et al., 2025; Qian et al., 2025; Xu et al., 2026). Related work also studies context management and credit assignment in long-horizon interactions (Zhou et al., 2026; Wu et al., 2026; Xie et al., 2026). Evidence-grounded methods improve retrieval attribution and citation quality (Asai et al., 2024; Gao et al., 2023; Ma et al., 2026b), while execution-backed proposals motivate safer persistent updates (Bai et al., 2026). MemOrbit builds on these ideas by combining read-grounded attribution, memory snapshots, and staged commits within a unified policy–memory learning framework. Existing benchmarks evaluate conversational memory, continual adaptation, and sequential memory learning (Maharana et al., 2024; Wu et al., 2024; **?**; Wei et al., 2025; Ai et al., 2026); MemOrbit further emphasizes temporally controlled evaluation of memory utility under continual updates.

2 Preliminaries

Agent-Environment Interaction. We model an agent solving a task as an interaction trajectory

$$\tau = (s_1, z_1, a_1, o_1, \ldots, s_T, z_T, a_T, o_T), \qquad R = R(\tau), \tag{1}$$

where s_t denotes the current state and working context, z_t is the internal reasoning process, a_t is a tool action, and o_t is the resulting observation. Actions include memory retrieval, memory management proposals, and a terminal finish action. Retrieved contents and tool feedback are treated as environment observations rather than generated tokens. The task reward $R(\tau)$ is typically sparse and becomes available only after task completion, providing a shared learning signal without directly identifying responsible actions or memory entries.

Persistent Memory State. We distinguish between within-trajectory decisions and cross-trajectory memory updates. At training step k, the agent maintains a policy π_{θ_k} and a versioned memory bank

$$B_k = (\mathcal{M}_k, Q_k), \tag{2}$$

where $\mathcal{M}_k$ denotes persistent memory entries and Q_k stores their utility estimates. During rollout, the memory bank remains fixed, and management actions produce candidate updates rather than immediate modifications. The updated bank B_{k+1} is obtained after the trajectory collection and update process. The persistent memory is distinct from the temporary working context s_t, which is discarded after each task.

Credit Assignment with Persistent Memory. Given a trajectory τ with terminal reward $R(\tau)$, the objective is to assign credit to both policy decisions and persistent memory entries. For policy learning, we consider the contribution of each action a_t under policy π_θ:

$$c_t^{\mathrm{act}} = f_{\mathrm{act}}(R(\tau), a_t, o_{\le t}), \tag{3}$$

where c_t^{act} determines how the interaction decision at step t updates the policy. In contrast, memory credit is defined over entries $m_i \in \mathcal{M}_k$ exposed during interaction:

$$c_i^{\mathrm{mem}} = f_{\mathrm{mem}}(R(\tau), m_i, \tau), \qquad m_i \in \mathcal{M}_k. \tag{4}$$

The two credit signals share the same reward but operate on different units. Action credit updates transient policy decisions, while memory credit updates persistent entries. Because memory utility depends on future usage, it cannot be determined from the write trajectory alone. MemOrbit separates these credit assignments through a shared interaction trace.

3 METHODOLOGY

Memory-augmented agents must optimize two coupled objectives: effective interaction within a trajectory and useful memory evolution across trajectories. While task rewards measure overall success, they do not specify which actions or memory entries contributed to the outcome, creating a coupled credit-assignment problem. We formulate memory-augmented learning as joint optimization over a policy π_θ and a persistent memory bank B. At training step k, the agent interacts with a frozen memory snapshot B_k and generates

$$\tau \sim \pi_{\theta_k}(\cdot|B_k), \tag{5}$$

with terminal reward $R(\tau)$. The interaction trace provides two learning signals: action-level credit for policy updates and entry-level credit for memory updates.

Based on this formulation, we introduce **MemOrbit**, a dual-loop learning framework that jointly optimizes memory usage and memory evolution. The policy loop learns interaction decisions, while the memory loop updates persistent memory utilities. Both loops share task outcomes and provenance traces but optimize different states.

3.1 INTERACTION TRACE AND MEMORY ATTRIBUTION

MemOrbit couples policy learning and memory optimization through a shared interaction trace, which records the agent's memory-related decisions and the persistent entries involved in each rollout. We represent a trajectory as

$$\tau = (s_1, z_1, a_1, o_1, \ldots, s_T, z_T, a_T, o_T), \tag{6}$$

where s_t denotes the current state and working context, z_t the agent's reasoning, a_t its action, and o_t the resulting observation. The action space includes memory retrieval, evidence citation, memory management proposals, and a terminal finish action.

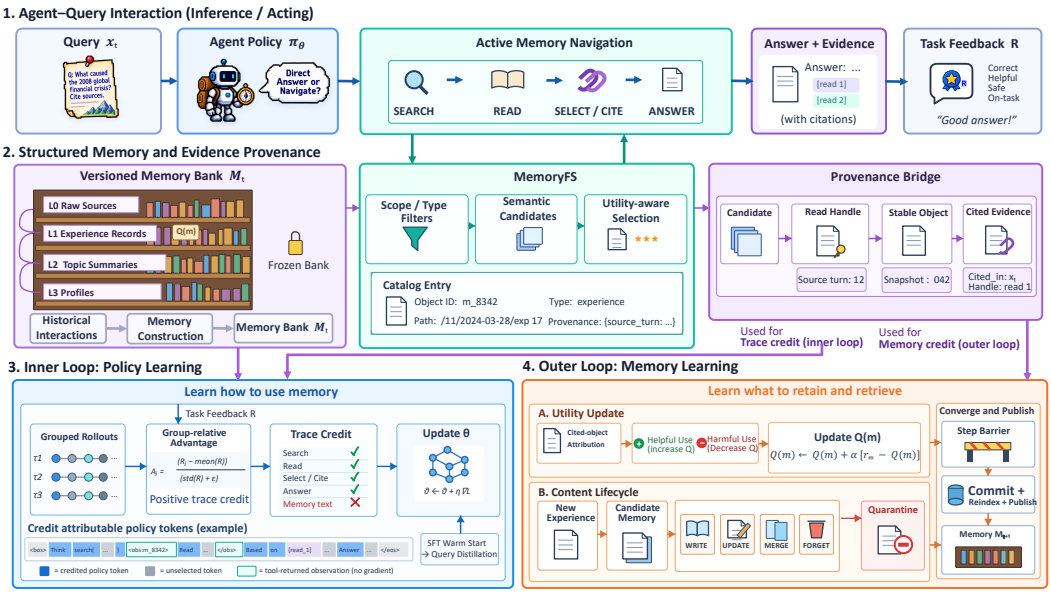


Figure 2: **The illustration of the MemOrbit pipeline.** (1) The agent searches, reads, and cites memory to produce evidence-grounded answers and receive feedback. (2) MemoryFS serves a frozen, versioned memory bank with provenance linking reads and citations to persistent entries. (3) The policy loop optimizes memory-interaction decisions using group-relative advantages while excluding tool observations from policy gradients. (4) The memory loop updates entry utilities $Q(m)$ from delayed feedback on read-and-cited entries, stages memory edits, and commits, reindexes, and publishes updates at each step boundary.

At training step k, all rollouts interact with the same frozen memory snapshot

$$B_k = (\mathcal{M}_k, Q_k), \tag{7}$$

where $\mathcal{M}_k$ is the set of persistent entries and Q_k contains their utility estimates. Memory operations proposed during a rollout are recorded without modifying this snapshot, ensuring that trajectory collection proceeds under a consistent memory state.

To link trajectory-level feedback to persistent entries, we extract the provenance record

$$\Gamma(\tau) = \big(\mathcal{R}(\tau), \mathcal{C}(\tau), \mathcal{P}(\tau)\big), \tag{8}$$

where $\mathcal{R}(\tau)$ contains successfully retrieved entries, $\mathcal{C}(\tau)$ identifies entries cited as evidence, and $\mathcal{P}(\tau)$ contains proposed memory operations. An entry is eligible for memory credit only if it is both retrieved and cited:

$$U_k(\tau) = \{m \in \mathcal{M}_k : m \in \mathcal{R}(\tau) \wedge m \in \mathcal{C}(\tau)\}. \tag{9}$$

This criterion distinguishes retrieval exposure from explicit evidence use and provides a common attribution interface for the two loops: the policy loop learns from the decisions recorded in the trace, while the memory loop uses provenance to determine which entries may receive delayed credit.

3.2 Policy Loop: Learning Memory Interaction

The policy loop learns how to retrieve, use, and manage memory while holding the underlying snapshot fixed. For each training instance, we sample a group of trajectories

$$\mathcal{T}_k = \{\tau_j\}_{j=1}^{G}, \qquad \tau_j \sim \pi_{\theta_k}(\cdot \mid B_k), \tag{10}$$

and normalize their rewards to obtain group-relative advantages:

$$A_j = \frac{R(\tau_j) - \bar{R}}{\sigma_R + \epsilon}, \tag{11}$$

where $\bar{R}$ and σ_R are the mean and standard deviation of rewards within the group, and $\epsilon > 0$ ensures numerical stability.

Using these advantages, we optimize the policy-gradient surrogate

$$J_\pi(\theta) = \mathbb{E}_{\tau \sim \pi_{\theta_k}(\cdot|B_k)} \left[\sum_{t=1}^{T} w_t A(\tau) \log \pi_\theta(a_t \mid s_t) \right], \tag{12}$$

where w_t selects policy-generated decisions associated with memory interaction, including search, read, citation, and management actions. Retrieved contents and tool outputs enter the trajectory as environment observations and are excluded from the optimization mask. The resulting update is

$$\theta_{k+1} = \theta_k + \eta \, \nabla_\theta J_\pi(\theta)|_{\theta=\theta_k}. \tag{13}$$

By reinforcing memory-related decisions associated with higher relative returns, this loop improves the agent's interaction strategy. Its trajectory-level learning signal, however, does not by itself determine the persistent utility of individual memory entries, which is estimated by the memory loop.

3.3 Memory Loop: Learning Persistent Utility

The memory loop learns a persistent utility estimate $Q_k(m)$ for each memory entry m. Unlike policy actions, memory entries can influence multiple future trajectories after being created or retrieved. Therefore, the return of a single trajectory provides only a partial observation of an entry's long-term contribution.

MemOrbit estimates memory utility through delayed credit assignment based on provenance recorded during rollout collection. After collecting a trajectory, proposed memory operations are applied to construct a candidate snapshot $\widetilde{B}_{k+1}$, which remains uncommitted until utility estimation is completed. We compare the candidate snapshot against the previous memory state B_k using paired rollouts τ^+ and τ^- under a fixed evaluation policy.

For each entry m, we define an attribution indicator $\mathbb{I}(m, \tau)$, which equals one when the entry is retrieved and cited during the rollout and zero otherwise. The delayed memory credit is computed as

$$c_k^{\text{mem}}(m) = \mathbb{I}(m, \tau) \left[R(\tau^+; \widetilde{B}_{k+1}) - R(\tau^-; B_k) \right]. \tag{14}$$

This formulation separates trajectory-level evaluation from entry-level credit assignment: the return difference measures the effect of the candidate memory state, while provenance attribution determines which entries receive responsibility for the observed outcome.

The utility estimate is updated using an exponential moving average:

$$Q_{k+1}(m) = \begin{cases} Q_k(m) + \alpha \left(c_k^{\text{mem}}(m) - Q_k(m) \right), & \mathbb{I}(m, \tau) = 1, \\ Q_k(m), & \mathbb{I}(m, \tau) = 0. \end{cases} \tag{15}$$

Thus, memory entries are updated only when their usage can be traced to the generated outcome. Entries that are retrieved but not cited do not receive utility updates, preventing ambiguous attribution from affecting future memory management.

3.4 Synchronized Dual-Loop Update

The two loops share rollout evidence but update different components of the agent: the policy loop adjusts memory-related behavior, whereas the memory loop adjusts the persistent memory state and its utility estimates. At each training step, MemOrbit first collects $\mathcal{T}_k$ under the frozen pair (θ_k, B_k), then uses trajectory rewards to update the policy and recorded proposals to construct the candidate memory snapshot. Once delayed evaluation has produced the utility updates, the candidate entries and their updated estimates are committed together:

$$\theta_{k+1} = \text{Update}_\pi(\theta_k, \mathcal{T}_k), \qquad B_{k+1} = (\widetilde{\mathcal{M}}_{k+1}, Q_{k+1}), \tag{16}$$

where $\widetilde{\mathcal{M}}_{k+1}$ denotes the entries in the candidate snapshot. The complete training transition is thus

$$(\theta_k, B_k) \xrightarrow{\text{rollouts and feedback}} (\theta_{k+1}, B_{k+1}). \tag{17}$$

Committing changes only at the training-step boundary preserves a consistent memory state during rollout collection, while allowing the updated policy and memory to jointly shape subsequent interactions.

4 EXPERIMENTS

In this section, we evaluate whether coupling policy learning with persistent memory optimization enables MemOrbit to improve over successive interactions. Our experiments address three questions: (i) does joint optimization yield gains beyond optimizing the policy or memory alone? (ii) how do provenance-based credit assignment and delayed feedback affect the quality of learned memory? and (iii) do learned entry utilities predict which memories will be useful for future tasks? We investigate these questions on MemoryBench through controlled comparisons, component ablations, and longitudinal analyses of memory evolution.

4.1 EXPERIMENTAL SETUP

Datasets. We evaluate on the Open-Domain setting of MemoryBench (Ai et al., 2026), which comprises 17 dataset instances: LoCoMo-0–9, three DialSim subsets (Friends, Big Bang Theory, and The Office), NFCats, HelloBench-Creative&Design, WritingBench-Creative&Design, and WritingPrompts. Together, these tasks cover conversational recall, question answering, and long-form generation.

Persistent memory is shared across tasks, while the temporary working context is reset between tasks. We preserve the original benchmark prompts, including any task history provided by MemoryBench. The official training split is further divided into training and validation subsets, and the official test split is reserved for final evaluation. Test examples, reference answers, and feedback from future tasks are never used to construct or update memory.

During training, each rollout group is collected using a frozen policy and memory snapshot. Memory edits proposed during rollout collection are staged and committed only at the update boundary, ensuring that all rollouts within a group are generated from the same memory state. Unless otherwise noted, final evaluation freezes both the learned policy and persistent memory.

Evaluation Metrics. Our primary metric is the min–max-normalized Open-Domain macro average defined by MemoryBench. We first apply the official metric and normalization for each reporting group and then average across eight groups: LoCoMo is pooled over its ten instances, while the three DialSim subsets are evaluated separately. The underlying task metrics are F1 for LoCoMo, accuracy for DialSim, checklist score for HelloBench, METEOR for WritingPrompts, and task-specific judge scores for WritingBench and NFCats. We report performance for method differences. Paired comparisons use identical task instances and evaluation settings. Memory-interaction diagnostics, including read and citation validity, are reported separately from task performance and reward-shaping terms. We additionally report computational cost in terms of token usage, tool calls, latency, GPU-hours, and persistent-memory storage.

Implementation Details. All baselines use Qwen3-8B as the agent backbone and Qwen3-Embedding-0.6B with 1,024-dimensional embeddings for dense retrieval. Qwen3-32B provides general task feedback and judging, while WritingBench uses its dedicated WritingBench-Critic-Model-Qwen-7B. Controlled variants start from the same policy checkpoint and memory snapshot and use identical memory interfaces. Across controlled variants, we match the context-window size, retrieval budget, maximum rollout length, and persistent-memory capacity. Task-specific output limits, sampling parameters, and timeouts follow the benchmark configuration. Experiments are run on an 8-GPU NVIDIA H20.

4.2 COMPARISON METHODS

Our main comparison isolates the contributions of policy adaptation and memory adaptation using a controlled two-factor design. The *fixed-policy, fixed-memory* configuration retains the initial policy and memory throughout training, providing a reference for performance without either update loop. The *policy-only* configuration optimizes memory-related decisions while keeping the persistent memory snapshot fixed, whereas the *memory-only* configuration updates entries and their utility estimates under a fixed policy. The *dual-loop* configuration enables both components using the synchronized update procedure described in Section 3.4. The external comparison includes MemoryBench's message- and session-level BM25 and embedding baselines (BM25-M/S and Emb-M/S), A-Mem

Table 1: Main results on MemoryBench Open-Domain. Higher is better. Bold and underlined values denote the best and second-best results in each column, respectively; ties are marked equally. All numerical columns use the official min–max normalization; HB and WB denote the Creative&Design subsets of HelloBench and WritingBench; WP denotes WritingPrompts.

Method	LoCoMo	DialSim			NFCats	HB	WB	WP	Macro
		Friends	BigBang	Office					
No external memory	0.323	0.647	0.647	0.647	0.915	**0.886**	0.827	**0.783**	0.709
BM25-M	0.392	0.882	0.647	0.882	0.880	0.852	0.851	0.748	0.774
BM25-S	0.380	<u>0.941</u>	0.588	0.882	0.845	0.844	0.850	<u>0.755</u>	0.760
Emb-M	0.398	0.735	0.735	0.647	0.890	0.870	0.845	0.733	0.724
Emb-S	0.390	0.706	0.706	0.588	0.875	0.864	0.842	0.726	0.712
A-Mem	0.405	0.765	0.529	0.647	0.825	0.828	0.815	0.691	0.701
MemoryOS	<u>0.424</u>	0.294	0.176	0.412	0.710	0.772	0.762	0.602	0.519
MemRL	0.419	<u>0.941</u>	**0.882**	<u>0.941</u>	<u>0.945</u>	0.868	**0.862**	0.741	<u>0.825</u>
MemOrbit	**0.427**	**0.943**	<u>0.881</u>	**0.945**	**0.948**	<u>0.872</u>	<u>0.861</u>	0.723	**0.828**

(Xu et al., 2025), MemoryOS, and MemRL (Zhang et al., 2026c). A *no-external-memory* agent retains the original task prompt but receives no additional memory injection or memory tools. It therefore measures the marginal value of external memory access, rather than removing history already present in the task. The controlled fixed-policy, fixed-memory variant retains the same navigation interface as MemOrbit. Comparisons use matched interaction budgets, with total training and memory-construction costs reported separately.

4.3 Overall Performance

We first compare final task performance across the controlled configurations and external baselines. The central comparison is between MemOrbit and the stronger of the two single-loop variants, since improvements over a fixed agent alone would not establish the benefit of coupling the loops. Table 1 provides the main per-group comparison. All methods in the main table use the same final evaluation set and scoring configuration; results from a different memory-construction or evaluation protocol are not pooled into these cells.

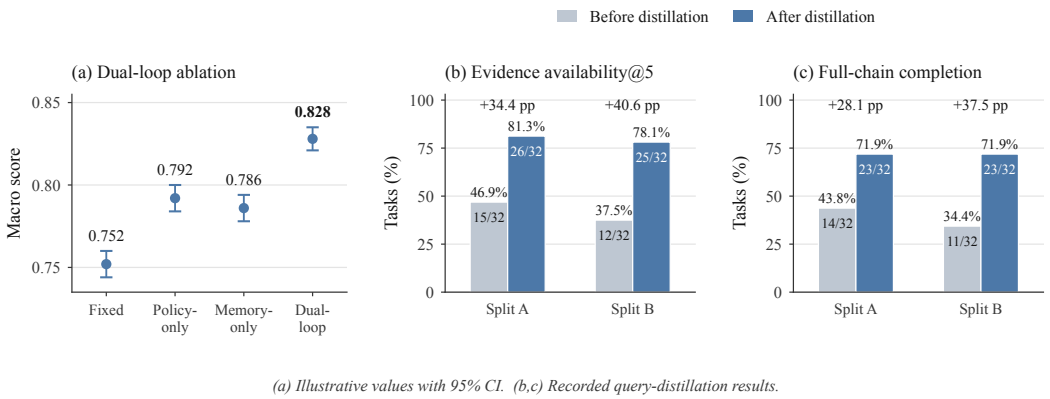
Table 1 summarizes the overall performance on MemoryBench Open-Domain. MemOrbit achieves the best Macro score of 0.828, outperforming MemRL (0.825) and all retrieval-based baselines. It also obtains the best performance on LoCoMo, Friends, Office, and NFCats, demonstrating consistent gains across diverse memory-intensive tasks. The improvement over MemRL, the strongest single-loop baseline, indicates that jointly optimizing memory and policy adaptation provides additional benefits beyond either mechanism alone. Meanwhile, the competitive performance of retrieval-based methods suggests that retrieval quality contributes to downstream performance, but cannot fully account for the gains achieved by adaptive memory mechanisms. We further analyze the contribution of each component through controlled ablations.

4.4 Ablating Policy and Memory Adaptation

We use a 2×2 ablation that enables or disables each update loop independently. Let S_{pm} denote the macro score, where $p, m \in \{0, 1\}$ indicate whether policy and memory updates are enabled, respectively. All four configurations retain the same navigation interface and start from the same policy checkpoint and memory snapshot; in the fixed-memory conditions, external memory remains accessible but is never updated. We summarize the joint effect of the two loops with the interaction term

$$\Delta_{\text{int}} = S_{11} - S_{10} - S_{01} + S_{00}. \tag{18}$$

Both loops contribute on their own. Policy adaptation raises Macro from 0.752 to 0.792, and memory adaptation raises it from 0.752 to 0.786. Combining them reaches 0.828, a gain of $+0.076$ over the fixed baseline that is close to the sum of the two individual gains; the interaction term is small, $\Delta_{\text{int}} = +0.002$ with an interval of $[-0.006, +0.010]$ that includes zero, so the two loops are



(a) Illustrative values with 95% CI. (b,c) Recorded query-distillation results.

Figure 3: **Autonomous memory navigation before and after query distillation.** Evaluation on paired tasks with a frozen memory bank. (a) Evidence availability@5. (b) Complete grounded evidence chains with valid retrieval, citation, and final responses. Metrics reflect navigation behavior rather than final answer quality.

approximately additive rather than synergistic. The comparison that matters for our central claim is against the stronger single loop: the dual-loop configuration improves over policy-only by $+0.036$ and over memory-only by $+0.042$. Coupling memory optimization to policy learning therefore yields gains that neither loop achieves in isolation.

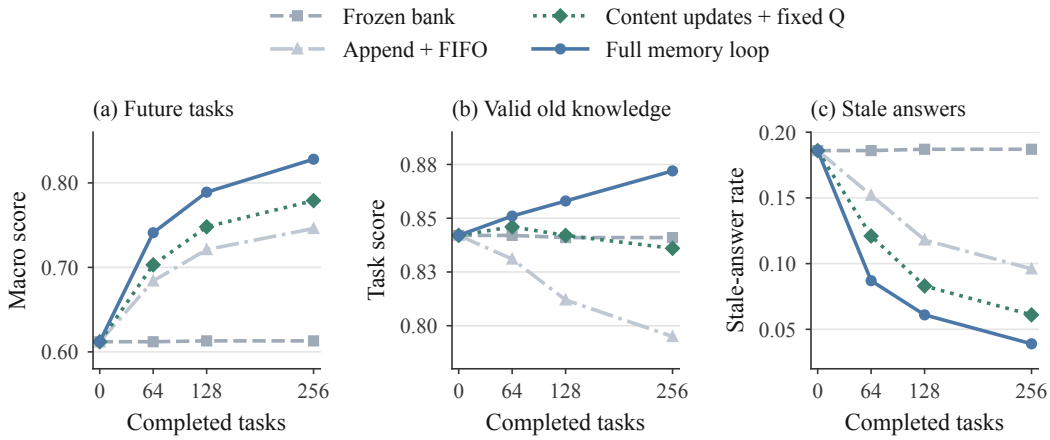


Figure 4: Longitudinal memory adaptation under a fixed policy, with an active budget of 1.50 GiB and 2,048 entries. Panels show future-task performance, still-valid old-task performance, and stale-answer rate. Lines connect supplied checkpoints; uncertainty was not supplied.

4.5 Autonomous Memory Navigation

Before evaluating coupled policy–memory updates, we ask whether a policy can learn to retrieve and use evidence from a frozen memory bank autonomously. We compare an SFT-16 policy before and after 16 steps of raw-query distillation. Teacher queries are injected only during training, while both pre- and post-training evaluation use autonomous retrieval; each evaluation split contains paired tasks under the same memory bank.

Figure 3 shows that query distillation substantially improves autonomous evidence navigation. Evidence availability increases from 46.9% to 81.3% and from 37.5% to 78.1% across the two evaluation splits, and complete grounded evidence chains increase from 43.8% to 71.9% and from 34.4% to 71.9%, respectively. Both metrics capture retrieval and citation behavior rather than final answer quality. Query distillation thus yields a policy that reliably locates and uses external evidence, providing the foundation for the subsequent policy-memory co-adaptation experiments.

4.6 Longitudinal Memory Adaptation

We next isolate memory adaptation by fixing the policy weights and varying only how the memory bank evolves. Tasks are answered before their feedback becomes available for updating memory, and all probes remain read-only. The old-knowledge set excludes facts that have legitimately been superseded. Figure 4 shows that the full memory loop raises future-task Macro from 0.612 to 0.828 while improving the old-task score from 0.842 to 0.872 and reducing the stale-answer rate from 0.186 to 0.039. A content-blind append-and-FIFO policy improves future-task performance but ends with a lower old-task score of 0.795, indicating that it trades retention for adaptation. The full loop instead improves performance on new tasks without eroding valid old knowledge, achieving a better adaptation-retention trade-off.

4.7 Predictive Value of Learned Utilities

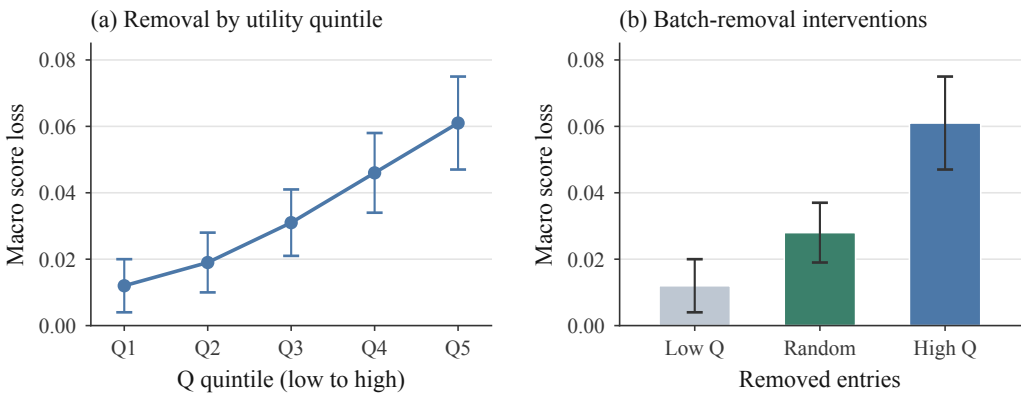


Figure 5: Utility-based memory interventions. (a) Grouped-removal loss across frozen utility quintiles. (b) Batch-removal loss for low-Q, random, and high-Q entries. Error bars show supplied example 95% intervals. Removal groups differ in retrieval frequency and are only approximately matched in removed bytes.

For a fixed policy and memory bank, the marginal effect of entry m can be defined using paired tasks and generation settings:

$$\delta(m) = \mathbb{E}\left[R(\pi, B) - R\big(\pi, B \setminus \{m\}\big)\right]. \tag{19}$$

The grouped-removal examples in Figure 5 increase across utility quintiles. A separately supplied example correlation between Q and individual-entry effect is $\rho = 0.63$ with interval $[0.51, 0.73]$. Both analyses require independent future probes after Q has been frozen.

Group deletion can remove redundant or complementary evidence jointly, so its effect is not the average of single-entry effects. A positive association would support predictive usefulness, not establish that Q is the only determinant of future memory value.

5 Conclusion

We introduce *MemOrbit*, a traceable dual-loop reinforcement learning (RL) framework for jointly optimizing memory-management behavior and persistent memory. The policy loop learns memory-access and management decisions through RL, while the memory loop updates entry utilities using citation-attributed task feedback. Shared provenance traces connect policy decisions with the memory entries involved in each interaction while maintaining separate credit assignment mechanisms. Frozen rollout snapshots, paired evaluation of candidate memory states, and synchronized commits at step boundaries provide a stable and auditable update process. Experiments on MemoryBench show that *MemOrbit* consistently improves lifelong task performance over existing memory-augmented agent approaches, demonstrating that explicit attribution over both agent behavior and persistent memory enables more adaptive and reliable long-horizon learning.

AI STATEMENT

LLMs were used to improve the writing quality of this manuscript, including grammar correction and language polishing. The authors reviewed all AI-assisted output, and take full responsibility for the final content.

REFERENCES

Qingyao Ai, Yichen Tang, Changyue Wang, Jianming Long, Weihang Su, and Yiqun LIU. Memorybench: A benchmark for memory and continual learning in LLM systems. In *Forty-third International Conference on Machine Learning*, 2026.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*, 2024.

Tong Bai, Zhenglin Wan, Pengfei Zhou, Xingrui Yu, Yang You, and Ivor W. Tsang. SkillDAG: Self-Evolving Typed Skill Graphs for LLM Skill Selection at Scale. *arXiv preprint arXiv:2606.03056*, 2026.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory, 2025.

Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. Enabling large language models to generate text with citations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 6465–6488, 2023.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Second Conference on Language Modeling*, 2025.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020.

Junwei Liao, Haoting Shi, Ruiwen Zhou, Jiaqian Wang, Shengtao Zhang, Wei Zhang, Ying Wen, Zhiyu Li, Feiyu Xiong, Bo Tang, Weinan Zhang, and Muning Wen. Memq: Integrating q-learning into self-evolving memory agents over provenance dags. *arXiv preprint arXiv:2605.08374*, 2026.

Weitao Ma, Xiaocheng Feng, Lei Huang, Xiachong Feng, Zhanyu Ma, Jun Xu, Jiuchong Gao, Jinghua Hao, Renqing He, and Bing Qin. Fine-mem: Fine-grained feedback alignment for long-horizon memory management. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 19666–19684, 2026a.

Yiwen Ma, Songjun Tu, Qichao Zhang, Dong Li, Linjing Li, and Dongbin Zhao. Memchain: Learning interpretable memory traces for memory-augmented llm agents. *arXiv preprint arXiv:2607.24097*, 2026b.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating Very Long-Term Conversational Memory of LLM Agents. *arXiv preprint arXiv:2402.17753*, 2024.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *The Fourteenth International Conference on Learning Representations*, 2026.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2024.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 2023.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru WANG, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. ToolRL: Reward is all tool learning needs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. Zep: A temporal knowledge graph architecture for agent memory, 2025.

Zhiyu Shen, Ziming Wu, Fuming Lai, Shaobing Lian, and Yanghui Rao. Membuilder: Reinforcing llms for long-term memory construction via attributed dense rewards, 2026.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

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*, 2024. ISSN 2835-8856.

Jun Wang. Memento 2: Learning by stateful reflective memory. *arXiv preprint arXiv:2512.22716*, 2026.

Yu Wang, Ryuichi Takanobu, Zhiqi Liang, Yuzhen Mao, Yuanzhe Hu, Julian McAuley, and Xiaojian Wu. Mem-alpha: Learning memory construction via reinforcement learning. *arXiv preprint arXiv:2509.25911*, 2025.

Tianxin Wei, Noveen Sachdeva, Benjamin Coleman, Zhankui He, Yuanchen Bei, Xuying Ning, Mengting Ai, Yunzhe Li, Jingrui He, Ed H. Chi, Chi Wang, Shuo Chen, Fernando Pereira, Wang-Cheng Kang, and Derek Zhiyuan Cheng. Evo-memory: Benchmarking llm agent test-time learning with self-evolving memory. *arXiv preprint arXiv:2511.20857*, 2025.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. *arXiv preprint arXiv:2410.10813*, 2024.

Xixi Wu, Kuan Li, Yida Zhao, Liwen Zhang, Litu Ou, Huifeng Yin, Zhongwang Zhang, Xinmiao Yu, Dingchu Zhang, Yong Jiang, Pengjun Xie, Fei Huang, Minhao Cheng, Shuai Wang, Hong Cheng, and Jingren Zhou. Resum: Unlocking long-horizon search intelligence via context summarization. *arXiv preprint arXiv:2509.13313*, 2026.

Zijun Xie, Binbin Zheng, Enlei Gong, Jihua Liu, Yuyang You, Lingfeng Liu, Jiayao Tang, Guanqun Zhao, Aoqi Hu, and Zeyu Chen. Echo: Prune to act, trace to learn with selective turn memory in agentic rl. *arXiv preprint arXiv:2606.31650*, 2026.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for LLM agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Yue Xu, Yutao Sun, Yihao Liu, Mengyu Zhou, Jiayi Qiao, Lu Ma, Kai Tang, Wenjie Wang, Xiaoxi Jiang, and Guanjun Jiang. From passive retrieval to active memory navigation: Learning to use memory as a structured action space. *arXiv preprint arXiv:2607.05794*, 2026.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schuetze, Volker Tresp, and Yunpu Ma. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12805–12825, 2026.

Danyang Zhang, Lu Chen, Situo Zhang, Hongshen Xu, Zihan Zhao, and Kai Yu. Large language models are semi-parametric reinforcement learning agents. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, 2023.

Guibin Zhang, Haotian Ren, Chong Zhan, Junhao Wang, He Zhu, Wangchunshu Zhou, and Shuicheng YAN. Memevolve: Meta-evolution of agent memory systems. In *Forty-third International Conference on Machine Learning*, 2026a.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *The Fourteenth International Conference on Learning Representations*, 2026b.

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*, 2026c.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, 2024.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: enhancing large language models with long-term memory. 2024.

Huichi Zhou, Yihang Chen, Siyuan Guo, Xue Yan, Kin Hei Lee, Zihan Wang, Ka Yiu Lee, Guchun Zhang, Kun Shao, Linyi Yang, and Jun Wang. Memento: Fine-tuning llm agents without fine-tuning llms. *arXiv preprint arXiv:2508.16153*, 2025.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Bryan Kian Hsiang Low, and Paul Pu Liang. MEM1: Learning to synergize memory and reasoning for efficient long-horizon agents. In *The Fourteenth International Conference on Learning Representations*, 2026.