MEMEFF: ENHANCING SELF-EVOLVING AGENTS THROUGH BEHAVIORAL EFFICACY-GUIDED MEMORY OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning from past experiences to optimize the memory of large language models (LLMs) has emerged as a key paradigm for autonomous agents handling interactive real-world tasks. However, existing studies focus on leveraging external feedback to improve memory utility, overlooking the influence of memory on LLM agent behavior. This oversight makes it difficult to measure memory's contribution to task performance, leading to suboptimal memory guidance and hindering the further improvement of LLM agents. To bridge this gap, we propose **MemEff**, a runtime learning framework that explicitly integrates the behavioral efficacy of accumulated memory into the self-evolution process of LLM-based agents. MemEff quantifies behavioral efficacy by measuring the difference in action generation log-probabilities between agents with and without memory guidance. By combining this measure with environmental feedback, MemEff estimates the potential performance gains of memories from historical interactions and identifies high-utility memories to guide LLM agents. Extensive experiments demonstrate that MemEff enhances the utility of retrieved memories and achieves state-of-the-art average performance across multiple benchmarks. These results highlight the importance of behavioral efficacy in memory optimization and continual self-evolution of LLM agents. Codes and data are available at `https://anonymous.4open.science/r/MemEff-75CD`.

1 INTRODUCTION

Large Language Models (LLMs) exhibit remarkable capabilities in context comprehension and reasoning, inspiring widespread efforts to develop autonomous agents to handle interactive, real-world tasks Wang et al. (2024b). However, the static nature of LLMs limits the agents' abilities to adapt to dynamic environments Xi et al. (2025a). To overcome this limitation, recent research has explored self-evolving LLM agents that continuously improve their capabilities by learning from past experiences Gao et al. (2026). As a key component of LLM agents, memory enables the accumulation and reuse of knowledge from historical interactions Zhang et al. (2025), making memory optimization a promising paradigm for building self-evolving agents.

Existing optimization methods primarily leverage external feedback to improve memory utility. For example, AWM Wang et al. (2025), CER Liu et al. (2025) and ReasoningBank Ouyang et al. (2026) distill reusable knowledge from successful or failed task trajectories, while MemRL Zhang et al. (2026) and JitRL Li et al. (2026b) use reward signals to guide memory utility estimation and optimize agent policies. These methods generally assume that memories appearing in high-reward trajectories have high utility, with the rewards provided by the environment or LLM evaluation.

Despite notable advancements, these methods overlook the influence of memory on LLM agent behavior, resulting in suboptimal memory selection for task guidance Xiong et al. (2026). To quantify this impact, we introduce **Behavioral Efficacy**, which measures the difference in action generation log-probabilities between agents with and without memory guidance, as formalized in Equation 6. As shown in Figure 1, high-reward memories consistently exhibit lower behavioral efficacy than the overall average. These findings expose a misalignment between external reward signals and the actual task guidance of LLM agents. Furthermore, our empirical analysis uncovers a non-monotonic

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.

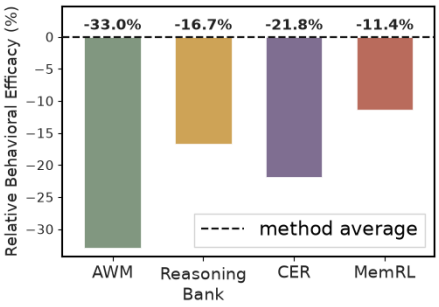


Figure 1: Relative behavioral efficacy of high-reward (≥ 0.8) memories.

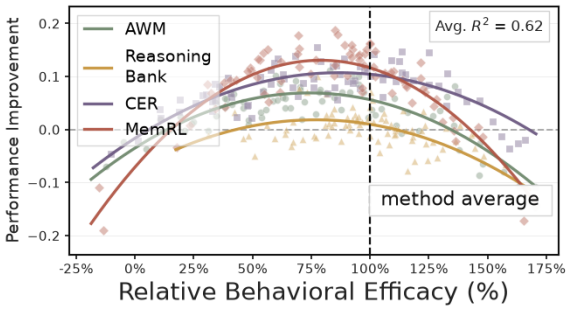


Figure 2: Association between behavioral efficacy and performance improvement.

association between behavioral efficacy and task performance: *moderate behavioral efficacy leads to the highest performance improvement*, whereas both weak and strong influence tend to compromise performance, as illustrated in Figure 2. This phenomenon indicates that current methods struggle to identify memories that effectively enhance agent capabilities through guidance, highlighting the critical role of behavioral efficacy in memory optimization.

To bridge this gap, we propose **MemEff**, a runtime learning framework that explicitly integrates the behavioral efficacy of accumulated memory into the self-evolution process of LLM-based agents. MemEff operates via two stages: *1) Efficacy-guided Memory Induction:* Distills agent-environment interaction trajectories into structured memories, retaining those that align with successful patterns while diverging from failures cases based on task outcomes and behavioral efficacy. *2) Utility-Aware Memory Selection:* Predicts the memory utility by modeling the association between historical efficacy and performance improvements, then selects the memory with the highest utility to guide subsequent action generation. Through iterative task execution, MemEff accumulates effective memories and refines the agent's policy, enabling LLM agents to continually self-evolve at test time.

We conduct experiments on three multi-turn interactive tasks, WebArena-Lite Zhou et al. (2024), AlfWorld Shridhar et al. (2021) and ScienceWorld Wang et al. (2022). Extensive experiments demonstrate that MemEff achieves state-of-the-art performance, outperforming several strong baselines with average success rates of 58.77% on WebArena-Lite, 94.45% on AlfWorld and an average reward of 87.27 on ScienceWorld. Further analysis confirms that MemEff significantly enhances the utility of retrieved memories, with a performance margin of 23.95% over semantic-based retrieval. Notably, it alleviates the mismatch between behavioral efficacy and performance improvement.

Our main contributions are summarized as follows:

- We highlight the significance of behavioral efficacy in LLM agent memory by empirically demonstrating its non-monotonic relationship with task performance improvements.
- We propose MemEff, a runtime learning framework that assesses and incorporates the behavioral efficacy of agent memory into the self-evolution of LLM-based agents.
- Extensive experiments demonstrate the effectiveness of MemEff, which achieves state-of-the-art average task performance and significantly enhances the utility of retrieved memories.

2 Related Work

Self-Evolving LLM Agents. LLM-based agents can iteratively perceive, act, and interact with dynamic environments to accomplish complex tasks Wooldridge & Jennings (1995); Wang et al. (2024b); Xi et al. (2025a). However, the static nature of LLM parameters fundamentally limits the improvement of agent adaptability, motivating the development of self-evolving LLM agents Gao et al. (2026). In previous works, agents learn from self-generated interaction trajectories by updating model parameters via supervised fine-tuning or reinforcement learning Sun et al. (2026); Xi et al. (2025b); Qi et al. (2025), which often incurs substantial computational and data costs due to repeated LLM training. Consequently, recent studies have shifted focus toward optimizing the memory modules to enhance LLM agent capabilities without modifying the underlying model Luo et al. (2026). By distilling experience from agents' own behaviors, our approach efficiently reuses memory to handle subsequent tasks, thereby ensuring sustained improvement Pritzel et al. (2017).

Memory Optimization in LLM Agents. Memory provides a lightweight mechanism for LLM agents to retain and exploit experience Zhang et al. (2025). Existing memory optimization methods extract reusable episodic memories Shu et al. (2026) or valuable skills Wang et al. (2024a) from interaction trajectories to guide agents on similar tasks. While prior studies Xu et al. (2026); Zhong et al. (2024); Zheng et al. (2024) typically rely on substantial offline data for memory construction, recent work has begun exploring test-time optimization to adapt memories from online interactions. For instance, AWM Wang et al. (2025), ReasoningBank Ouyang et al. (2026), and CER Liu et al. (2025) establish a runtime feedback loop between memory refinement and task execution, but they rely solely on iterative experience distillation from trajectories without explicitly guaranteeing memory utility. In contrast, JitRL Li et al. (2026b) and MemRL Zhang et al. (2026) start to develop mechanisms to evaluate memory utility via LLM-based judgment or environmental rewards. However, these approaches focus on external feedback, overlooking the intrinsic impact of memory content on agent behavior. Overall, MemEff optimizes memory retrieval by predicting utility based on behavioral efficacy, facilitating the selection of memories that effectively guide agent behavior.

3 PRELIMINARIES

In this section, we formulate memory-augmented LLM generation as a Memory-Utility Markov Decision Process (MU-MDP), defined by the tuple $(\mathcal{U}, \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{M})$, where $\mathcal{U}$ is the instruction space, $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the state transition function, $\mathcal{R}$ is the task reward function, and $\mathcal{M}$ is the evolving memory space. Given a task $u \in \mathcal{U}$ and a memory item $m \in \mathcal{M}$, we define the memory utility U as the performance improvement and assume the existence of functional mapping f with behavioral efficacy:

$$U(u, m) = r(u, m) - r_0(u) = f\left(D(u, m)\right), \tag{1}$$

where $r_0(u)$ and $r(u, m)$ denote rewards for memory-free and memory-augmented agents, respectively, and $D(u, m)$ characterizes the influence of memory m on the agent's behavior.

At each step t, the agent receives state s_t and leverages $\mathcal{M}_t$ to generate an action a_t. The joint policy $\pi(a_t \mid u, s_t, \mathcal{M}_t)$ is formulated as the marginal probability over all possible retrieved items m:

$$\pi(a_t \mid u, s_t, \mathcal{M}_t) = \sum_{m \in \mathcal{M}_t} \mu(m \mid u, s_t, \mathcal{M}_t) \cdot p_{\mathrm{LLM}}(a_t \mid u, s_t, m), \tag{2}$$

where $\mu(m \mid u, s_t, \mathcal{M}_t)$ is the **Retrieval Policy** for selecting memories, and $p_{\mathrm{LLM}}(a_t \mid u, s_t, m)$ is the **Inference Policy** parameterized by a frozen LLM. This approach transforms retrieval from a passive match into an active decision process. The executable action a_t is generated by the policy but is strictly conditioned on the retrieved content m. In LLM agent environments with deterministic state transition $\mathcal{T}$, the action determines the environmental feedback and next state s_{t+1}, which finally affects the task reward. Accordingly, the optimal retrieval policy μ^* can be defined by:

$$\mu^* = \arg\max_{\mu} \mathbb{E}_{m \sim \mu(\cdot \mid u, s_t, \mathcal{M}_t)} \left[r(u, m)\right]. \tag{3}$$

Since $r_0(u)$ is the memory-free task reward and independent of m, we have:

$$\mu^* = \arg\max_{\mu} \mathbb{E}_{m \sim \mu(\cdot \mid u, s_t, \mathcal{M}_t)} \left[r(u, m) - r_0(u)\right] = \arg\max_{\mu} \mathbb{E}_{m \sim \mu(\cdot \mid u, s_t, \mathcal{M}_t)} \left[U(u, m)\right]. \tag{4}$$

Combining Equation 1 and 4, the optimal retrieval policy maximizes the expectation of the behavioral efficacy function:

$$\mu^* = \arg\max_{\mu} \mathbb{E}_{m \sim \mu(\cdot \mid u, s_t, \mathcal{M}_t)} f\left[D(u, m)\right] \tag{5}$$

In contrast to prior studies Zhou et al. (2025); Zhang et al. (2026); Wang et al. (2026b), which define the optimization objective solely based on cumulative task reward, our MU-MDP formulation encourages the retrieval policy to identify memories that make meaningful contributions to the LLM agent's performance through behavioral efficacy.

4 METHODOLOGY

Building upon the MU-MDP formulation defined in Section 3, we propose MemEff, a framework that enables frozen LLMs to self-evolve through runtime learning from historical experiences. This section will begin by elaborating on the association between behavioral efficacy and agent performance improvement, serving as the motivation for our approach. We then describe the two-stage memory optimization process of MemEff. The comprehensive workflow is illustrated in Figure 3.

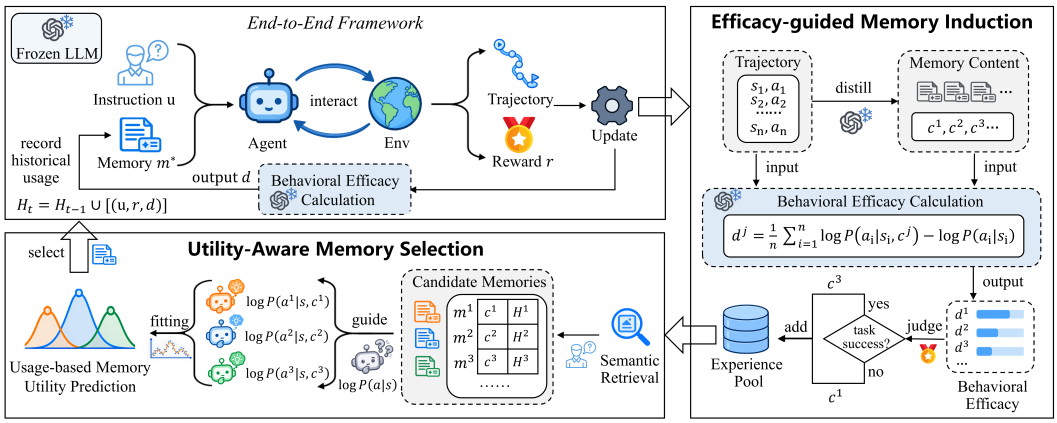


Figure 3: The overall framework of the proposed MemEff, which consists of two stages: Efficacy-guided Memory Induction and Utility-Aware Memory Selection.

4.1 Motivation of MemEff

Inspired by on-policy distillation Yang et al. (2026); Wu et al. (2026), we quantify the influence of memory on agent behavior by calculating the difference in log-probabilities of the same executed actions under memory-guided and memory-free conditions. Specifically, given a trajectory $\tau_n = (s_1, a_1, s_2, a_2, \ldots, s_n, a_n)$ and a memory textual content c, we define the **behavioral efficacy** as:

$$d(\tau_n, c) = \frac{1}{n}\sum_{t=1}^{n}[\log p_\theta(a_t \mid s_t, c) - \log p_\theta(a_t \mid s_t)], \tag{6}$$

where θ denotes the LLM parameters, s_t and a_t are the state and action at timestep t, respectively. In this work, behavioral efficacy denotes a signed scalar value.

Intuitively, a high-utility memory is supposed to provide useful information that steers the agent behavior toward successful task completion. Motivated by this intuition, we analyze existing memory optimization methods on web navigation Zhou et al. (2024) and household Shridhar et al. (2021) tasks. Using `Gemma-4-31B-it` Team (2026) and `Qwen-3.5-27B` Qwen Team (2026a), we calculate the behavioral efficacy of self-generated memories and record the corresponding task rewards from their guided agent trajectories, defining *high-reward* memories as those with an average reward ≥ 0.8. Figure 1 shows that high-reward memories optimized by different methods exhibit substantially lower behavioral efficacy than their respective method-level averages. This discrepancy reveals that high reward signals do not necessarily translate into effective behavioral guidance. For visualization, we report relative behavioral efficacy, divided by each method's average.

Furthermore, we investigate the relationship between the behavioral efficacy and agent performance improvement over the memory-free baseline. For each task, the memory-free baseline is estimated through averaging multiple independent trials conducted independently by LLM agents. Figure 2 illustrates this association with the fitted quadratic curves. And each scatter point represents the average over multiple task trajectories. We observe a non-monotonic relationship where memory-induced performance gains increase at moderate behavioral efficacy but decline at both insufficient and excessive levels. This indicates that suboptimal memories produced by existing methods limit agent capabilities by offering either inadequate guidance or disruptive interference. Table 8 reports low fitting errors of these curves, confirming this association phenomenon.

Our empirical analysis highlights that effective guidance is essential for enhancing agent task performance across LLM backbones, motivating our modeling of memory utility via behavioral efficacy.

4.2 Efficacy-guided Memory Induction

Self-evolving agents usually induce valuable experiences from self-generated trajectories, which are subsequently reused to guide their execution on similar tasks. However, the memory guidance does not necessarily translate into performance improvement. Recent studies have shown that self-generated experiences provide limited gains over no-experience baselines Li et al. (2026a); Wang

et al. (2026a). In particular, memory reuse could induce behavioral deviations, causing agents to over-imitate behaviors that were successful in previous trials Xiong et al. (2026), or neglecting corrections for failed steps Kruengkrai & Yoshino (2025).

To mitigate this issue, MemEff introduces the behavioral efficacy into memory induction to ensure that extracted memories faithfully reflect their impact on agent behavior. Given a task instruction u and its completed trajectory $\tau_n = (s_1, a_1, s_2, a_2, \ldots, s_n, a_n)$, the frozen LLM first generates w candidate memory content items $\mathcal{C} = \{c^1, \ldots, c^\omega\}$ by summarizing τ_n. Each candidate is prepended to the original trajectory to evaluate the behavioral efficacy d^i via Equation 6, replacing any existing memory content. The memory retention rule is defined as:

$$c^* = \begin{cases} \arg \max\limits_{c^i \in \mathcal{C}, i \in 1,\ldots,\omega} d^i, & r(u) \geq \delta_1, \\ \arg \min\limits_{c^i \in \mathcal{C}, i \in 1,\ldots,\omega}, d^i, & r(u) < \delta_1, \end{cases} \tag{7}$$

where c^* is the selected memory stored in the experience pool and δ_1 is manually defined threshold of task success. Our objective is to select one item per ω candidates, targeting maximal behavioral efficacy for successful trajectories to preserve effective execution patterns, while identifying those with minimal efficacy for failed trajectories to avoid erroneous actions.

To further expand the search space of candidate memories, we follow prior works Zhai et al. (2025); Liu et al. (2026) and iteratively refine the generated memories. Specifically, the previously generated memory content c_{i-1} is provided as an additional prompt input to the LLM, which is instructed to identify and address potential flaws in the current experience:

$$c^i = p_\theta \left(I, \tau, c^{i-1}\right), \quad i = 2, \ldots, w, \tag{8}$$

where c_1 is directly generated with trajectory τ, p_θ denotes the same LLM as the task-executing policy, and I represents the prompt for experience summarization, as detailed in Appendix D.4.

4.3 Utility-Aware Memory Selection

To optimize the retrieval policy, we predict memory utility before task execution. The structure of memory bank $\mathcal{M}$ is a set of memory usage history tuples:

$$\mathcal{M} = \left\{m^i = (c^i, H^i)\right\}_{i=1}^{|\mathcal{M}|}, \quad H^i = \left\{(u^j,\, r^j - r_0(u^j),\, d^j)\right\}_{j=1}^{|H^i|}, \tag{9}$$

where c^i is the summarized memory content, and H^i records the historical usage of memory. Each record in H^i contains the task instruction u^j, the corresponding performance improvement $r^j - r_0(u^j)$ over the memory-free baseline, and the behavioral efficacy d^j induced by the memory.

Conventional similarity-based retrieval assumes that semantically similar memories are likely to be useful for the current task. However, such retrieved experiences often generalize poorly to agentic tasks Xu et al. (2026); Xia et al. (2026), where their effectiveness depend on environment-specific factors beyond semantic similarity. Therefore, to identify high-utility memories, MemEff adopts a two-phase retrieval strategy. In the first phase, given a current instruction u, we construct a candidate memory set $\mathcal{M}_{\text{cand}}(u)$ by filtering the experience pool according to cosine similarity:

$$\mathcal{M}_{\text{cand}}(u) = \text{TopK}_k \left(\{m^i \mid \text{sim}\big(\text{Emb}(u), \text{Emb}(c^i)\big) > \delta_2\}\right), \tag{10}$$

where $\text{Emb}(\cdot)$ denotes the embedding model that maps raw text into a normalized vector representation, k is the number of retrieved memories, and δ_2 is a threshold for semantic similarity. If $\mathcal{M}_{\text{cand}}(u) = \emptyset$, MemEff relies solely on the frozen LLM for exploration.

In the second phase, we rank the k candidates via usage-based memory utility estimation. Specifically, for each candidate memory $m^i \in \mathcal{M}_{\text{cand}}(u)$, the LLM agent performs a lookahead rollout of the first q actions conditioned on its memory content. We also apply Equation 6 to compute the behavioral efficacy $\tilde{d}\big(\tau_q^i, c^i\big)$ over the first q actions between the memory-augmented and memory-free agents. In practice, the behavioral impact of the initial steps dominates the overall trajectory value, which is shown in Table 6. Therefore, we configure $q = 1$ with $\tau_1^i = (u, a_1^i)$, using the first-step efficacy as a proxy for the overall behavioral efficacy. MemEff then employs a nonparametric kernel regression Nadaraya (1964) over the usage history H^i to capture the non-monotonic association

observed in Section 4.1, without imposing a predefined functional form. We define two anisotropic Gaussian product kernels on behavioral efficacy and instruction embeddings:

$$k_d(\tilde{d}, d^j) = \exp\left(-\frac{(\tilde{d} - d^j)^2}{2\sigma_d^2}\right), \quad k_u(e_u, e_{u^j}) = \exp\left(-\frac{\|e_u - e_{u^j}\|_2^2}{2\sigma_u^2}\right), \tag{11}$$

where $e_u = \mathrm{Emb}(u)$ and $e_{u^j} = \mathrm{Emb}(u^j)$, the bandwidths σ_d and σ_u are set to the median pairwise distances within $\{d^j\}$ and $\{e_{u^j}\}$ in H^i, respectively. This design is motivated by their complementary roles in utility estimation, which is detailed in Appendix A.2. The predicted utility $\widehat{\Delta r}_i(u)$ of memory m^i for the current instruction u is computed as

$$\widehat{\Delta r}_i(u) = \frac{\sum_{j=1}^{|H_i|} w_{ij}\big(r^j - r_0(u^j)\big)}{\sum_{j=1}^{|H_i|} w_{ij}}, \quad w_{ij} = k_d\big(\tilde{d}(\tau_q^i, c^i), d^j\big)\, k_u(e_u, e_{u^j}). \tag{12}$$

We weight historical usages with respect to their instruction embeddings and behavioral efficacy for the current task, using their improvements $r^j - r_0(u^j)$ to estimate the reward gain. Finally, the agent interacts with the environment using the memory $m^{i^\star}$ of highest predicted utility:

$$i^\star = \arg \max_{i:\, m^i \in \mathcal{M}_{\mathrm{cand}}(u)} \widehat{\Delta r}_i(u). \tag{13}$$

Upon task completion, we record these execution data as a usage entry for the retrieved memory $m^{i^\star}$ to improve future utility predictions.

5 Experiments

5.1 Experimental Setup

Evaluation Benchmarks and Protocol. We evaluate in three multi-step agent-based environments:

- **WebArena-Lite** Zhou et al. (2024): A realistic web environment comprising multiple functional websites, including Shopping Admin, GitLab, Map, Reddit, and Shopping. It challenges agents to navigate complex DOM trees and execute sequential actions to fulfill user instructions.
- **ALFWorld** Shridhar et al. (2021): An embodied environment for interactive household tasks, comprising both Seen and Unseen task settings. It requires agents to solve household tasks by navigating between rooms, manipulating objects, and executing correct action sequences.
- **ScienceWorld** Wang et al. (2022): A text-based interactive environment designed to simulate scientific reasoning, also featuring Seen and Unseen settings. It evaluates agents' ability to solve scientific problems, covering topics such as object conductivity and basic chemical reactions.

Following prior works Li et al. (2026b); Zhang et al. (2026), we adopt a multi-trial sequential testing protocol, where each task is evaluated over five consecutive episodes to simulate realistic online deployment, reporting performance on the final episode. All agents are evaluated with the **Average Reward** from environment feedback; for WebArena-Lite and ALFWorld, where rewards are binary ($r \in \{0, 1\}$), this metric is equivalent to the **Success Rate**. All values are scaled by 100.

Baselines. We compare against several strong runtime memory optimization baselines. **(1) Static**: A memory-free agent with fixed configuration. **(2) AWM** Wang et al. (2025): An episodic memory approach that extracts reusable workflows from successful trajectories. **(3) ReasoningBank** Ouyang et al. (2026): A framework that distills generalizable reasoning strategies from both successful and failed experiences. **(4) CER** Liu et al. (2025): A training-free method that synthesizes past experiences into a dynamic memory buffer for contextual retrieval and scenario matching. **(5) JitRL** Li et al. (2026b): An inference-time framework that retrieves relevant trajectories to estimate action advantages for logit adjustment. **(6) MemRL** Zhang et al. (2026): A reinforcement learning framework that estimates expected rewards for episodic memories to guide memory retrieval.

Implementation Details. MemEff employs the closed-source model `GPT-5.1` Singh et al. (2025) and open-source model `Gemma-4-31B-it` Team (2026) as LLM backbones for the inference policy, and the LLM used for behavioral efficacy calculation is `Gemma-4-31B-it`. We set the task success threshold $\delta_1 = 0.7$, semantic retrieval threshold $\delta_2 = 0.6$ and candidate counts $\omega = k = 3$ for both memory induction and retrieval. Notably, CER and JitRL baselines were not implemented in ALFWorld and ScienceWorld. More implementation details are provided in Appendix D.

Table 1: Performance Comparison. The best and second best results for each method are in **bold** and underlined, respectively. "Avg." denotes the average performance within each benchmark.

Inference Model	Method	WebArena-Lite						ALFWorld			ScienceWorld		
		Admin	GitLab	Map	Reddit	Shopping	Avg.	Seen	Unseen	Avg.	Seen	Unseen	Avg.
GPT-5.1	Static	61.14	34.38	33.33	46.67	31.30	41.36	78.81	76.14	77.47	77.08	76.21	76.65
	AWM	54.86	36.25	42.96	68.57	34.35	47.40	83.73	75.57	79.65	80.12	79.90	80.01
	Reasoning Bank	58.29	33.75	42.22	58.10	31.30	44.73	81.34	78.57	79.96	81.32	77.65	79.49
	CER	60.00	41.25	**50.37**	63.81	29.57	49.00	-	-	-	-	-	-
	JitRL	61.71	44.38	47.41	67.62	35.22	51.27	-	-	-	-	-	-
	MemRL	**74.86**	48.13	44.44	59.05	26.09	50.51	90.45	89.43	89.94	84.15	82.66	83.41
	MemEff (Our)	73.71	**59.38**	48.15	**75.24**	**37.39**	**58.77**	**94.33**	**94.57**	**94.45**	**88.23**	**86.31**	**87.27**
Gemma-4-31B-it	Static	43.43	38.75	34.07	49.52	29.57	39.07	75.67	76.00	75.84	78.19	76.37	77.28
	AWM	52.00	39.38	39.26	63.81	29.13	44.71	81.49	76.57	79.03	78.34	78.08	78.21
	Reasoning Bank	57.71	45.63	40.00	65.71	23.91	46.59	77.76	81.72	79.74	79.72	80.43	80.08
	CER	58.86	36.25	43.70	67.62	30.87	47.46	-	-	-	-	-	-
	JitRL	68.00	35.63	37.78	48.57	30.43	44.08	-	-	-	-	-	-
	MemRL	65.14	42.50	40.00	71.43	27.83	49.38	89.40	89.57	89.49	82.12	81.61	81.87
	MemEff (Our)	**66.86**	**49.38**	**44.44**	**72.38**	**33.91**	**53.39**	**94.63**	**95.14**	**94.89**	**86.24**	**86.71**	**86.48**

5.2 Main Results

We report the average scores over five independent runs in Table 1. Our proposed MemEff achieves the superior average performance on all three benchmarks under both inference backbones.

On WebArena-Lite, MemEff achieves the highest average performance for both GPT-5.1 and Gemma-4-31B-it, reaching 58.77 and 53.39, respectively. It surpasses the strongest baselines, JitRL and MemRL, by 7.50 and 4.01 points. Compared to these baselines, which rely solely on external feedback signals for learning, our approach integrates behavioral efficacy into the memory optimization process, demonstrating its effectiveness in web environments. Performance improvements are particularly notable in complex state-dependent environments like Admin (+53.9% over Static with Gemma-4-31B-it) and Reddit (+61.2% over Static with GPT-5.1) due to the precise prediction of memory utility. In contrast, reward-based utility prediction may mislead in difficult tasks, evidenced by MemRL's marginal performance gains with GPT-5.1 on Shopping and Reddit websites.

On ALFWorld and ScienceWorld, MemEff achieves state-of-the-art performance in both seen and unseen tasks. GPT-5.1 obtains the average scores of 94.45 and 87.27 on ALFWorld and ScienceWorld, exceeding the top baseline by 4.51 and 3.86 points, respectively. Similar improvements are observed with Gemma-4-31B-it, with gains of 5.40 and 4.61 points, demonstrating the robustness of MemEff across different environments and both open-source and closed-source models.

5.3 In-Depth Analysis

In this subsection, we analyze MemEff in terms of: the behavioral impact of induced memories, retrieval effectiveness, iterative optimization process and generalizability across model backbones.

Memory Induction. To evaluate whether induced memories reinforce successful behaviors while avoid erroneous ones, we conduct a perturbation experiment with GPT-5.1. We compare memories from different methods by inserting or replacing them in original trajectories, prompting the agent to generate new rollouts via re-interaction with environments. The deviation between original and new action sequences from task trajectories is quantified using cosine similarity, normalized Levenshtein distance (NLD), and our behavioral efficacy. The first two metrics range from 0 to 1, with higher values indicating greater consistency, and higher behavioral efficacy indicates a stronger tendency to reproduce the corresponding trajectory. To ensure memory generalization, we adopt a cross-task setting in which memories extracted from one task are not applied to guide that same task, and introduce Reflexion Shinn et al. (2023) as a comparative baseline.

Table 2: Memory-induced deviation from original trajectories on WebArena-Lite.

Method	Success Trajectory			Failure Trajectory		
	Cosine Similarity↑	NLD↑	Behavioral Efficacy↑	Cosine Similarity↓	NLD↓	Behavioral Efficacy↓
Reflexion	0.53	0.44	-62.35	0.59	0.48	-56.28
AWM	0.68	0.61	-28.71	0.62	0.62	-31.94
ReasoningBank	0.62	0.58	-32.90	0.64	0.65	-32.48
CER	0.66	0.59	-34.77	0.56	0.47	-57.79
MemEff (Ours)	**0.71**	**0.73**	**-20.85**	**0.54**	**0.42**	**-61.73**

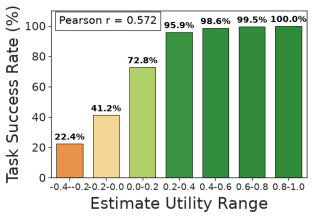


Figure 4: Task success rate versus utility range.

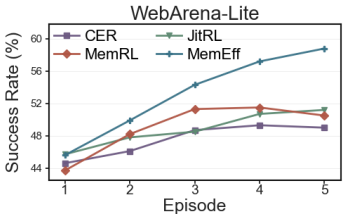


Figure 5: Performance changes over five episodes.

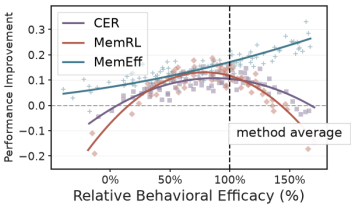


Figure 6: The association in MemEff.

Table 2 shows that different forms of memory induction differentially affect agent behavior. Reflexion causes strong deviations on both successful and failed trajectories, with NLD of 0.44 and 0.48 respectively, reflecting the strong influence of memories containing concrete trajectories. In contrast, AWM and ReasoningBank extract abstract experiences and exert weak behavioral influence on both types of trajectories, whereas CER displays stronger guidance through website-specific experiences. MemEff further effectively preserves successful behaviors while suppressing erroneous ones, achieving the highest semantic similarity of 0.71 on successful trajectories and the lowest behavioral efficacy of -61.73 on failed trajectories.

Retrieval Effectiveness. To validate the improved utility of retrieved memories, we compare MemEff's retrieval mechanism against semantic similarity and reward-based (MemRL) retrieval baselines. For each task, we record the experience pool state before retrieval, select different memories using each strategy, and measure their utility by the resulting improvement in task success rate. As shown in Table 3, MemEff achieves an average improvement of 23.95% over the "*Semantic*" baseline across two scenarios. Although the reward-based method MemRL uses environmental rewards to filter out memories with low expected rewards, MemEff further incorporates behavioral efficacy to enhance retrieval accuracy for the current task, leading to an additional 17.63% average boost in retrieval utility.

Table 3: MemEff utility improvement versus baseline retrieval methods (%).

	vs. Semantic	vs. MemRL
WebArena-Lite		
GPT-5.1	+41.40	+34.60
Gemma-4-31B-it	+24.13	+15.37
ALFWorld Unseen		
GPT-5.1	+18.03	+11.11
Gemma-4-31B-it	+12.24	+9.43

Figure 4 also shows a strong positive correlation between predictive utility and success rate in ALFWorld (Pearson $r = 0.572$), with success rates rising from 22.4% to 100% across confidence bins, confirming MemEff's effectiveness in ranking memories. Our retrieval mechanism mitigates the impact of inefficient memories in LLM self-evolution, which is detailed in Appendix B.3.

Iterative Optimization. To characterize the runtime learning, we analyze performance improvement with GPT-5.1 on WebArena-Lite across multiple optimization episodes, which is presented in Figure 5. It is observed that CER lacks mechanisms to evaluate the quality of induced memories, resulting in negligible performance gains across multiple iterations. Although JitRL and MemRL exhibit initial improvements in early iterations, followed by rapid performance saturation or even degradation in subsequent rounds. The iterative optimization of these baseline methods on the ALFWorld Unseen follows the same trend as described above, as shown in Figure 9. Conversely, MemEff demonstrates continual performance growth over five episodes in both scenarios.

Model Generalization. To evaluate the generalizability of MemEff, we employ GPT-5.1 as the inference model and employ three open-source models with different architectures, including Gemma-4-31B-it, Qwen3.5-27B, and Qwen3.8-27B Qwen Team (2026b), as scoring models for behavioral efficacy calculation. As shown in Table 4, MemEff achieves consistently strong performance across different scoring models, peaking at 77.06 and 75.24 on WebArena-Lite's Admin and Reddit scenarios, and 94.78 on ALFWorld Unseen. These results demonstrate the behavioral efficacy is robust to different models, enabling MemEff to optimize memory effectively across different scoring LLM backbones.

Table 4: Performance comparison across different scoring models.

Method	Admin	Reddit	ALFWorld Unseen
Static	61.14	46.67	76.14
AWM	54.86	68.57	75.57
Reasoning Bank	58.29	58.10	78.57
MemRL	74.86	59.05	89.43
MemEff *w/ Scoring:*			
Gemma-4-31B-it	73.71	**75.24**	94.57
Qwen3.5-27B	**77.06**	67.62	93.28
Qwen3.8-27B	74.71	69.52	**94.78**

Table 5: A case study on WebArena-Lite. We present two memories with different estimated utilities and their resulting agent executions. The memory selected by MemEff guided the agent to **Success**, whereas the one chosen by MemRL led to **Failure**.

Instruction	Memory Content	Utility Scores	Agent's Behavior
Tell me the number of reviews that our store received by far that mention term "best".	**Memory A**: - Skill: `fill("review_filter_id","keyword");` - Usage: Check how many reviews mention a particular keyword, and manage any matching reviews if present...	MemRL: 0.39 MemEff: 0.53	**Action:** `fill("Review","best")` **Observation:** 2 comments **Success**
	Memory B: - Skill: `fill("name_filter_id","keyword");` - Usage: Enter a keyword into the name filter input and search by product name, user name, or target brand name...	MemRL: 0.60 MemEff: 0.36	**Action:** `fill("Nickname","best")` **Observation:** 0 comments **Failure**

5.4 Qualitative Analysis

Visualization. Following the setup in Figure 2, we similarly visualize the utility association exhibited by MemEff-optimized memories, as shown in Figure 6. Compared to CER, MemRL exhibits a narrower distribution concentrated within the moderate efficacy range. However, both methods are constrained by the balance between behavioral efficacy and performance improvement. MemEff overcomes this limitation by leveraging utility prediction based on behavioral efficacy to sample high-utility memories. This is manifested as a positive correlation where higher behavioral efficacy translates into greater performance improvements, demonstrating that MemEff effectively translates memory-induced behavioral influence into task performance improvements.

Case Study. We present a case study in Table 5, illustrating how behavioral efficacy-based retrieval selects task-relevant memories. The user's instruction is to find the number of reviews containing the word "best". Both Memory A and Memory B offer viable guidance, with Memory A providing a more specific method for filtering reviews. However, Memory B covers simpler filtering tasks and achieves a higher average reward, leading MemRL to assign it a higher score ($0.60 > 0.39$) and causing the agent to incorrectly filter by username. In contrast, MemEff assigns Memory A a higher utility for the current task ($0.53 > 0.36$), enabling successful task execution.

Table 6: Proportion of behavioral efficacy in the first q steps (%).

Benchmarks	$q=1$	$q=2$	$q=3$
WebArena-Lite	50.52	70.77	86.10
ALFWorld	61.69	76.34	82.43
ScienceWorld	54.69	73.31	79.83

Table 7: Comparison of optimization costs.

Metric	Static	AWM	Reasoning Bank	CER	JitRL	MemRL	MemEff
Cost	$45.90	$66.10	$58.30	$57.20	$74.30	$63.30	$77.15

5.5 Cost Analysis

Although behavioral efficacy calculation in MemEff increases computational overhead, Table 6 shows that the initial steps guided by memory account for most of the behavioral efficacy along the trajectory. Leveraging this insight, we limit efficacy calculations to the first three steps during memory induction and the first step during memory selection. This approach ensures that MemEff's additional computational overhead scales linearly with the number of tasks. Table 7 compares the costs of single-run memory optimization for GPT-5.1 on WebArena-Lite. MemEff's remains within the same order of magnitude as other methods while significantly improving task performance. Analysis of the computational overhead and detailed cost calculations are provided in Appendix C.

6 Conclusion

In this work, we propose MemEff, a runtime learning framework that integrates behavioral efficacy into the self-evolution of LLM agents. By calculating efficacy during memory induction and retrieval, MemEff selects high-utility memories that effectively translate guidance into task contributions, ensuring continuous performance improvement without weight updates. Extensive experiments show that MemEff outperforms existing test-time optimization methods, achieving a 23.95% utility improvement over semantic-based retrieval. Notably, MemEff alleviates the mismatch between behavioral efficacy and performance improvement in LLM agent memory.

AI USE STATEMENT

In this work, we used generative AI tools for drafting and editing prose, including the method section, and for checking internal consistency of notation against the implementation. We have reviewed all AI-assisted text. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

All experiments in this work were conducted within simulated task environments, including web navigation on synthetic websites and embodied AI tasks in virtual simulations. Since our methodology does not involve interactions with real-world systems, live user data, or physical hardware, it poses no risks related to privacy breaches, security vulnerabilities, or physical safety concerns.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our work, we have submitted the complete implementation code for the methods described in Section 4 to an anonymous repository `https://anonymous.4open.science/r/MemEff-75CD.`, which includes all core modules and the full set of prompts used for instructing LLM-based agents. The three benchmark datasets involved in Section 5 are publicly available online. Furthermore, detailed implementation specifics for all baseline methods, are thoroughly documented in Appendix D to ensure fair and transparent comparison.

REFERENCES

Norman R Draper and Harry Smith. *Applied regression analysis*, volume 326. John Wiley & Sons, 1998.

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, et al. A survey of self-evolving agents: What, when, how, and where to evolve on the path to artificial super intelligence. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=CTr3bovS5F. Survey Certification.

Canasai Kruengkrai and Koichiro Yoshino. Teaching text agents to learn sequential decision making from failure. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 31619–31635, 2025.

Xiangyi Li, Yimin Liu, Wenbo Chen, Bingran You, Zonglin Di, Yifeng He, Shenghan Zheng, Kyoung Whan Choe, Jiankai Sun, Shuyi Wang, et al. Skillsbench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 2026a.

Yibo Li, Zijie Lin, Ailin Deng, Xuan Zhang, Yufei He, Shuo Ji, Tri Cao, and Bryan Hooi. Just-in-time reinforcement learning: Continual learning in LLM agents without gradient updates. In *Forty-third International Conference on Machine Learning*, 2026b. URL https://openreview.net/forum?id=pLvye0zHUC.

Huanxi Liu, Kun Hu, Qiang Wang, Yuanzhao Zhai, Feng Dawei, Bo Ding, and Huaimin Wang. Cope: A framework for optimizing coordination between planning and execution in llm-based agents. In *Forty-third International Conference on Machine Learning*, 2026.

Yitao Liu, Chenglei Si, Karthik R Narasimhan, and Shunyu Yao. Contextual experience replay for self-improvement of language agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14179–14198, 2025.

Jinghao Luo, Yuchen Tian, Chuxue Cao, Ziyang Luo, Hongzhan Lin, Kaixin Li, Chuyi Kong, Ruichao Yang, and Jing Ma. From storage to experience: A survey on the evolution of LLM agent memory mechanisms. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 41622–41652, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.2069. URL https://aclanthology.org/2026.findings-acl.2069/.

Elizbar A Nadaraya. On estimating regression. *Theory of Probability & Its Applications*, 9(1): 141–142, 1964.

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, et al. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *International Conference on Learning Representations*, volume 2026, pp. 94327–94354, 2026.

Alexander Pritzel, Benigno Uria, Sriram Srinivasan, Adria Puigdomenech Badia, Oriol Vinyals, Demis Hassabis, Daan Wierstra, and Charles Blundell. Neural episodic control. In *International conference on machine learning*, pp. 2827–2836. PMLR, 2017.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Jiadai Sun, Xinyue Yang, Yu Yang, Shuntian Yao, Wei Xu, et al. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. In *International Conference on Learning Representations*, volume 2025, pp. 79791–79821, 2025.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026a. URL https://qwen.ai/blog?id=qwen3.5.

Qwen Team. Qwen3.8-Max: A new bar for coding and cowork, August 2026b. URL https://qwen.ai/blog?id=qwen3.8.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J. Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

Yiheng Shu, Padmaja Jonnalagedda, Xiang Gao, Bernal Jimenez Gutierrez, Weijian Qi, Kamalika Das, Huan Sun, and Yu Su. Remem: Reasoning with episodic memory in language agent. In *International Conference on Learning Representations*, volume 2026, pp. 55958–55987, 2026.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Zeyi Sun, Ziyu Liu, Yuhang Zang, Yuhang Cao, Xiaoyi Dong, Tong Wu, Dahua Lin, and Jiaqi Wang. SEAgent: Self-evolving computer use agent with autonomous learning from experience. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=AvBgmOGsdQ.

Gemma Team. Gemma 4 technical report, 2026. URL https://arxiv.org/abs/2607.02770.

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. URL https://mlanthology.org/tmlr/2024/wang2024tmlr-voyager/.

Jiongxiao Wang, Qiaojing Yan, Yawei Wang, Yijun Tian, Soumya Smruti Mishra, Zhichao Xu, Megha Gandhi, Panpan Xu, and Lin Lee Cheong. Reinforcement learning for self-improving agent with skill library. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1529–1550, 2026a.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024b.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. ScienceWorld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11279–11298, 2022.

Tong Wang, Pei Xu, Shiyue Cao, Likun Yang, Daipeng Li, Jianbin Jiao, and Kaiqi Huang. Samem: State-aware memory as a fine-grained memory for llm agents in decision-making. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 14691–14710, 2026b.

Yuyao Wang, Zhongjian Zhang, Mo Chi, Kaichi Yu, Yuhan Li, Miao Peng, Bing Tong, Chen Zhang, Yan Zhou, and Jia Li. Evomembench: Benchmarking agent memory from a self-evolving perspective. *arXiv preprint arXiv:2605.18421*, 2026c.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63897–63911. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/wang25bx.html.

Tianxin Wei, Noveen Sachdeva, Benjamin Coleman, Zhankui He, Yuanchen Bei, Xuying Ning, Mengting Ai, Yunzhe Li, Jingrui He, Ed H Chi, et al. Evo-memory: Benchmarking llm agent test-time learning with self-evolving memory. *arXiv preprint arXiv:2511.20857*, 2025.

Michael Wooldridge and Nicholas R Jennings. Intelligent agents: Theory and practice. *The knowledge engineering review*, 10(2):115–152, 1995.

Jinyang Wu, Shuo Yang, Zhengxi Lu, Fan Zhang, Yuhao Shen, Lang Feng, Haoran Luo, Zheng Lian, Shuai Zhang, Zhengqi Wen, et al. Seed: Self-evolving on-policy distillation for agentic reinforcement learning. *arXiv preprint arXiv:2607.14777*, 2026.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *Science China information sciences*, 68(2):121101, 2025a.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Xin Guo, Dingwen Yang, Chenyang Liao, Wei He, et al. Agentgym: Evaluating and training large language model-based agents across diverse environments. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27914–27961, 2025b.

Menglin Xia, Xuchao Zhang, Shantanu Dixit, Paramaguru Harimurugan, Rujia Wang, Victor Rühle, Robert Sim, Chetan Bansal, and Saravan Rajmohan. Memora: A harmonic memory representation balancing abstraction and specificity. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=zSrvkj0ers.

Zidi Xiong, Yuping Lin, Wenya Xie, Pengfei He, Zirui Liu, Jiliang Tang, Himabindu Lakkaraju, and Zhen Xiang. How memory management impacts llm agents: An empirical study of experience-following behavior. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 623–645, 2026.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *Advances in Neural Information Processing Systems*, 38:17577–17604, 2026.

Shuo Yang, Jinyang Wu, Zhengxi Lu, Yuhao Shen, Fan Zhang, Lang Feng, Shuai Zhang, Haoran Luo, Zheng Lian, Zhengqi Wen, et al. Opid: On-policy skill distillation for agentic reinforcement learning. *arXiv preprint arXiv:2606.26790*, 2026.

Yuanzhao Zhai, Huanxi Liu, Zhuo Zhang, Tong Lin, Kele Xu, Cheng Yang, Dawei Feng, Bo Ding, and Huaimin Wang. Empowering large language model agent through step-level self-critique and self-training. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1444–1454, 2025.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Zhuo Li, Yujie Zheng, Weinan Zhang, Ying Wen, Zhiyu Li, et al. Memrl: Self-evolving agents via runtime reinforcement learning on episodic memory. *arXiv preprint arXiv:2601.03192*, 2026.

Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model-based agents. *ACM Transactions on Information Systems*, 43(6):1–47, 2025.

Longtao Zheng, Rundong Wang, Xinrun Wang, and Bo An. Synapse: Trajectory-as-exemplar prompting with memory for computer control. In *International Conference on Learning Representations*, volume 2024, pp. 19036–19066, 2024.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: enhancing large language models with long-term memory. 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*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i17.29946. URL https://doi.org/10.1609/aaai.v38i17.29946.

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

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, volume 2024, pp. 15585–15606, 2024.

APPENDIX

The appendix of this paper is organized as follows.

- Appendix A presents motivational experiments, including an analysis of the relationship between behavioral efficacy and performance improvement, as well as utility prediction using our Gaussian kernels.
- Appendix B presents additional experimental results, including ablation studies, performance changes, and the proportion of inefficient memories during iterative optimization.
- Appendix C analyzes the computational complexity and practical costs of MemEff.
- Appendix D provides detailed experimental settings, including dataset descriptions, model configurations, baseline implementations, and hyperparameters. It also reports evaluation reliability and additional implementation details.

A MOTIVATIONAL EXPERIMENT

A.1 CORRELATION VALIDATION

In Section 4.1, our empirical study quantified the behavioral efficacy of memories across multiple task trajectories using Equation 6 and examined its association with performance improvement, as illustrated in Figure 2. Each scatter point represents the average over trajectories within a behavioral efficacy interval, with relative behavioral efficacy normalized by the mean behavioral efficacy of memories generated by the corresponding method.

Table 8: The quadratic fitting quality of different memory methods.

Method	$R^2 \uparrow$	RMSE $\downarrow$	MAE $\downarrow$
AWM	0.6973	0.0262	0.0207
Reasoning Bank	0.4874	0.0292	0.0227
CER	0.6401	0.0268	0.0215
MemRL	0.6590	0.0292	0.0240
MemEff	0.7794	0.0274	0.0214

We fitted a quadratic curve to these aggregated data points and evaluated the fitting precision using the Coefficient of Determination R^2, Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) Draper & Smith (1998). The detailed results are presented in Table 8. Most methods exhibit a high degree of fitting quality, confirming the robustness and validity of this association. Notably, Reasoning Bank, despite having the lowest R^2 of 0.4874, still maintains minimal prediction errors with an RMSE of 0.0292 and an MAE of 0.0227, indicating that the fitted curve remains close to the actual data points.

A.2 UTILITY PREDICTION ANALYSIS

To validate the predictive relevance of behavioral efficacy and instruction similarity for memory utility, we conduct a utility prediction experiment on held-out data. Each memory usage instance is represented by its behavioral efficacy, user instruction, and actual utility derived from environmental rewards. We evaluate whether these two factors can accurately predict the utility of an unseen memory usage. For each memory m, we collect a history of memory usages

$$H^m = \left\{ \left(u^j, d^j, \Delta r^j \right) \right\}_{j=1}^{|H^m|}, \quad \Delta r^j = r^j - r_0(u^j) \tag{14}$$

where u^j denotes the instruction, d^j denotes the behavioral efficacy of the memory, and Δr^j denotes the difference between the memory-guided task performance and the no-memory baseline.

We perform leave-one-out prediction within each memory history. For a held-out usage j, its utility Δr^j is predicted from the remaining usages $\ell \neq j$:

$$\widehat{\Delta r}^{\,j} = \frac{\sum_{\ell \neq j} K(x^j, x^\ell) \Delta r^\ell}{\sum_{\ell \neq j} K(x^j, x^\ell)}, \tag{15}$$

where x^j denotes the behavioral efficacy and instruction representation of usage j.

To isolate the contribution of each dimension, we consider three kernel configurations:

$$k_d(x^j, x^\ell) = \exp\left(-\frac{(d^j - d^\ell)^2}{2\sigma_d^2}\right), \quad k_u(x^j, x^\ell) = \exp\left(-\frac{\|e_{u^j} - e_{u^\ell}\|_2^2}{2\sigma_u^2}\right), \tag{16}$$

$$k_{du}(x^j, x^\ell) = k_d(x^j, x^\ell) k_u(x^j, x^\ell). \tag{17}$$

Here, $e_{u^j} = \mathrm{Emb}(u^j)$ and $e_{u^\ell} = \mathrm{Emb}(u^\ell)$ denote the instruction embeddings. The bandwidths σ_d and σ_u are fixed using the median pairwise distance within each corresponding history.

We evaluate prediction quality using cross-validated $\mathrm{MAE}_{\mathrm{cross}}$:

$$\mathrm{MAE}_{\mathrm{cross}} = \frac{1}{N}\sum_{j=1}^{N}\left|\Delta r^j - \widehat{\Delta r}^{\,j}\right|. \tag{18}$$

A lower MAE indicates more accurate utility prediction on held-out usages. We introduce the **Reward Mean** as a baseline, defined as the mean task reward over all other usages of the memory. We further report the *WinRate* metric, which measures the fraction of held-out usages for which the predicted utility is closer to the ground-truth utility than the Reward Mean baseline:

$$\mathrm{WinRate} = \frac{1}{N}\sum_{j=1}^{N}\mathbb{I}\left[\left|\Delta r^j - \widehat{\Delta r}^{\,j}\right| < \left|\Delta r^j - \widehat{\Delta r}^{\,j}_{\mathrm{avg}}\right|\right]. \tag{19}$$

As shown in Table 9, relying solely on behavioral efficacy (k_d) underperforms the Reward Mean baseline, with a WinRate of only 47.65%. In contrast, incorporating instruction similarity via k_u enhances utility prediction, reducing the MAE to 0.2003 and achieving a WinRate of 62.34%. Combining both kernels further reduces the MAE to 0.1487 and boosts the WinRate to 72.55%, demonstrating that behavioral efficacy and instruction similarity provide complementary signals for prediction. Furthermore, Table 3 confirms that our utility prediction method enables more effective memory selection. Overall, these results support the rationale and effectiveness of using Gaussian kernels to model memory utility.

Table 9: Memory utility prediction using different kernel dimensions.

Setting	$\mathrm{MAE}_{cross}\downarrow$	WinRate(%) $\uparrow$
Reward Mean	0.2458	-
Kernel k_d	0.2581	47.65
Kernel k_u	0.2003	62.34
Kernel k_{du}	**0.1487**	**72.55**

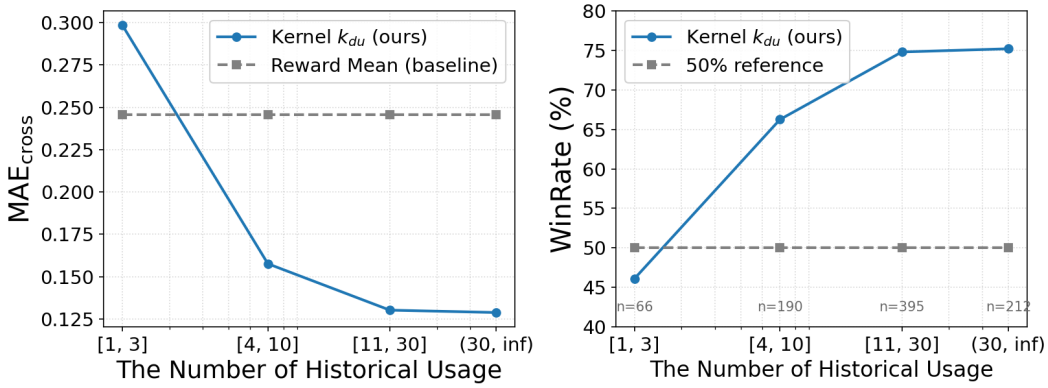


Figure 7: Prediction error (left) and WinRate (right) of our kernel estimator k_{du} versus the number of historical usages $|H_i|$.

Cold-start analysis. Extending the experiments' setup, we bin held-out usages by $|H_i| \in \{[1,3],[4,10],[11,30],(30,\infty)\}$ and compute $\mathrm{MAE}_{\mathrm{cross}}$ and $\mathrm{WinRate}$ per bin. Figure 7 shows that our kernel estimator underperforms the Reward Mean baseline in the cold-start bin ($|H_i| \leq 3$) in terms of both $\mathrm{MAE}_{\mathrm{cross}}$ and WinRate. Performance rebounds quickly once $|H_i| \geq 4$ and gradually stabilizes when $|H_i| > 30$.

B Additional Experiment

B.1 Ablation Studies

To analyze the sensitivity of MemEff to its two components, utility-guided memory induction and utility-aware memory retrieval, we evaluate the impact of the number of candidate memories for induction ω and the number of candidate memories for retrieval k on the Admin and Reddit tasks in WebArena-Lite, and on the ALFWorld Unseen dataset. Here, $\omega = 1$ indicates the absence of the induction module, and $k = 1$ signifies retrieval based solely on semantic similarity, without the involvement of behavioral efficacy.

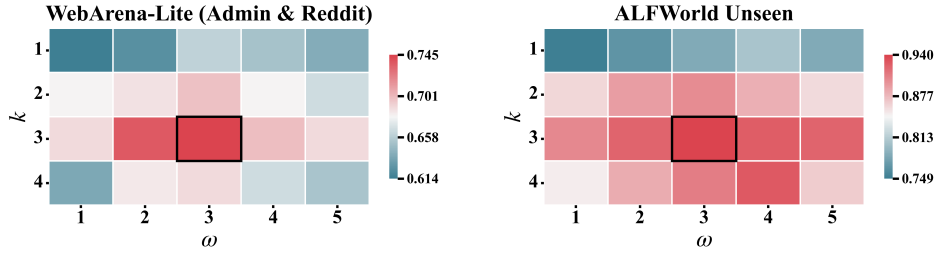


Figure 8: Impact of memory candidate counts ω and k.

As shown in Figure 8, the LLM agent achieves optimal performance when $\omega = 3$ and $k = 3$. Notably, k has a greater impact on agent performance than ω. This can be attributed to the fact that ω only affects the experience pool, whereas k determines which memories directly guide the agent's task execution. A small k value limits exploration of the memory space, while a large k value incorporates many rarely used memories. Since the utility estimates for these memories are unstable, they introduce noise into the agent's memory optimization, leading to reduced task performance.

B.2 Iteration Optimization

Figure 9 illustrates the performance changes over five iterations of memory optimization. Similar to CER in Figure 5, AWM and Reasoning Bank achieve only marginal performance gains through iterative optimization, as it lacks mechanisms to ensure the utility of self-generated experiences and relies primarily on accumulating memories. In contrast, MemEff and MemRL achieve faster and greater improvements. However, without accounting for the influence of memories on agent behavior, MemRL's performance begins to decline in the fourth iteration.

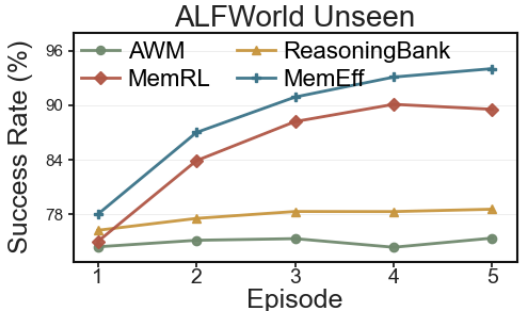


Figure 9: Performance changes through multiple iterations in the ALFWorld Unseen.

B.3 Memory Efficiency Analysis

To evaluate whether MemEff effectively mitigates the impact of inefficient memories, we analyze the proportion of retrieved memories with utility greater than 0 across five evolutionary rounds. Additionally, we introduced an ablation baseline, **MemEff w/ Reward Prediction**, which replaces the original utility prediction module with task reward prediction. Figure 10 illustrates the proportion of retrieved memories with positive task utility (> 0) over five rounds of iterative optimization.

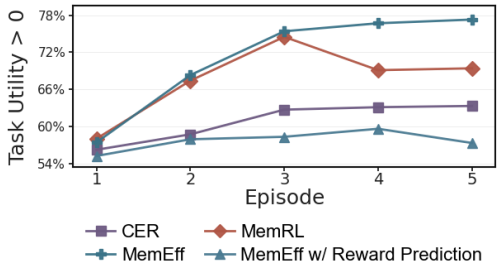


Figure 10: Proportion of retrieved memories with positive task utility (> 0) during iterative optimization on WebArena-Lite.

Without an explicit mechanism for ensuring memory utility, CER only gradually accumulates task-relevant memories throughout the optimization process. In contrast, MemRL estimates memory utility via environmental task rewards to guide retrieval, leading to an initial increase in the proportion of beneficial memories. However, because it fails to account for the intrinsic impact of memories on agent behavior, its performance degrades in later optimization rounds. Furthermore, given the lack of a significant association between task rewards and behavioral efficacy, the MemEff w/ Reward Prediction baseline cannot accurately identify high-utility memories through reward prediction. Conversely, MemEff ensures a sustained increase in the proportion of high-utility memories, demonstrating the effectiveness of direct utility modeling.

C Cost Analysis

This subsection analyzes the computational overhead introduced by behavioral efficacy calculation. For a dataset with n tasks, let T_1 and T_2 denote the average time for LLM generation and environment interaction respectively, with an average trajectory length of L. The baseline execution time is given by $T_{base} = O(nLT_1 + nLT_2)$. MemEff incurs additional costs of kT_1 for retrieval, which evaluates behavioral efficacy at the first step, and $3\omega T_1$ for induction, which evaluates efficacy over the first three steps. Thus the total complexity is expressed as:

$$T_{total} = O(nLT_1 + nLT_2 + nkT_1 + 3n\omega T_1) \tag{20}$$

k and ω are constants relative to the trajectory length L. Therefore the total complexity simplifies to $T_{total} = O(n(L + k + 3\omega))$, indicating that MemEff maintains the same asymptotic complexity as other baseline methods.

Table 7 presents the cost of single run memory optimization for our evaluation. The operational costs of MemEff and other training-free memory optimization methods are estimated based on their token consumption through the OpenRouter API, which provides a unified billing standard for these inference-based methods. Additionally, MemEff employs a locally deployed open-source model to compute behavioral efficacy. This process requires approximately 14.6 GPU-hours on an NVIDIA A800 (80GB) per optimization round. Using the public rental price of \$0.825 per hour from AutoDL [1], we estimate the additional cost to be around \$12.05 per round.

[1] https://www.autodl.com/home

D EXPERIMENTAL DETAILS

D.1 DATASETS AND MODELS

Our evaluation covers three challenging agent benchmarks, where agents must perceive changes in the environment state and interact with the environment through a constrained action space. Each task requires the agent to execute a sequence of actions over multiple steps to fulfill a user instruction, after which the environment evaluates both the resulting environment state and the agent's final response to determine the task reward. Table 10 summarizes the statistics of the three datasets, including the number of user task instructions, executable actions, interactive objects, and average steps required to complete each task. The large number of executable actions and interactive objects makes effective multi-step reasoning particularly challenging. Following prior work Li et al. (2026b); Zhang et al. (2026), we execute the task instructions sequentially, allowing each agent to accumulate reusable experiences through the corresponding memory optimization method.

Table 10: Statistics of the evaluation datasets.

Dataset	# Instruction	# Actions	# Objects	# Average Steps
Webarena-Lite	161	12	300+	7.71
AlfWorld	274	11	73	7.97
ScienceWorld	405	25	195	11.76

# denotes the number of this item.

WebArena-Lite Zhou et al. (2024) requires agents to interact with realistic web environments by parsing the Document Object Model (DOM) tree, identifying interactive elements (e.g., buttons, input fields, and links), and executing precise actions such as clicking and typing. It covers diverse scenarios, including e-commerce, collaborative development, and forum management. The environment provides a binary reward of 0 or 1, indicating whether the task is successfully completed.

AlfWorld Shridhar et al. (2021) requires agents to interpret high-level user instructions and complete household tasks by navigating between rooms, manipulating objects, and executing appropriate action sequences. In addition to seen tasks, AlfWorld includes unseen tasks to evaluate the agent's generalization ability. It also adopts a binary reward of 0 or 1 based on task completion.

ScienceWorld Wang et al. (2022) simulates scientific reasoning scenarios based on elementary school science curricula, covering tasks such as testing object conductivity and performing basic chemical reactions. It contains both seen and unseen tasks and provides a dense reward in $[0, 1]$ to measure the degree of task completion.

Table 11: Open-source models used in our experiments and their official sources.

Type	**Name**	**Source**
LLM	Gemma-4-31B-it	`https://huggingface.co/google/gemma-4-31b-it`
	Qwen3.5-27B	`https://huggingface.co/Qwen/Qwen3.5-27B`
	Qwen3.8-27B	`https://huggingface.co/Qwen/Qwen3.8-27B`
Embedding	all-MiniLM-L6-v2	`https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2`

All open-source models used in our experiments and their official sources are listed in Table 11. `Gemma-4-31B-it`, `Qwen3.5-27B`, and `Qwen3.8-27B` are used as the backbone models for agent inference or behavioral efficacy calculation. The embedding model `all-MiniLM-L6-v2` is used for all semantic similarity computations, including the first-stage semantic retrieval and the instruction-semantic Gaussian kernel k_u in the second-stage utility prediction.

Table 12: Hyperparameter settings for all memory optimization methods.

Method	Hyperparameter	WebArena-Lite	ALFWorld & ScienceWorld
AWM Wang et al. (2025)	LLM Temperature	0.0	0.0
	Retrieval Content Number	-	3
Reasoning Bank Ouyang et al. (2026)	LLM Temperature	0.8	0.8
	Retrieval Content Number	1	1
	MaTTS Number	5	5
CER Liu et al. (2025)	LLM Temperature	0.8	-
	Retrieval Content Number	5	-
JitRL Li et al. (2026b)	LLM Temperature	0.8	-
	Retrieval Neighbor Number	10	-
	Candidate Action Number	3	-
	Discount Factor	0.1	-
MemRL Zhang et al. (2026)	LLM Temperature	0.8	0.8
	Retrieval Content Number	5	5
	Candidate Memory Number	3	3
	Utility Learning Rate	0.3	0.3
MemEff	LLM Temperature	0.8	0.8
	Retrieval Content Number	5	5
	Candidate Memory Number	3	3
	Induction Memory Number	3	3

- denotes that the hyperparameter is not applicable or not implemented in the corresponding scenario.

D.2 Method Settings

For WebArena-Lite, we implement all baselines following the hyperparameter configurations specified in their original papers and evaluate them under the same test-time memory optimization protocol. For MemEff and the MemRL baseline, we adopt the memory induction strategy of CER Liu et al. (2025), which summarizes website-specific dynamic usages and functional skills from agent trajectories. All baseline methods learn exclusively from experiences collected during test-time interactions, without access to pre-collected offline data.

For ALFWorld and ScienceWorld, CER and JitRL do not provide implementations for these environments in their original works, and their memory methods are tightly coupled with specific scenarios. For AWM and ReasoningBank, we follow the corresponding baseline implementations in Evo-Memory Wei et al. (2025) and EvoMemBench Wang et al. (2026c), respectively, reusing their prompts and code to ensure consistency with prior evaluations. MemEff adopts the same memory induction prompts as MemRL.

Table 12 summarizes the hyperparameter configurations used for all memory optimization methods. The settings cover the key components of each method, including LLM sampling temperature, memory retrieval, test-time scaling, candidate action or memory selection, and utility estimation. Specifically, for single memories that contain multiple entries, retrieval-based methods use different budgets: 3 retrieved items for AWM, 1 for ReasoningBank, 5 for CER, and 5 for both MemRL and MemEff. JitRL retrieves 10 neighboring memories and considers 3 candidate actions for its reinforcement learning procedure. ReasoningBank uses a MaTTS value of 5, generating five test-time trajectories during memory induction. MemRL and MemEff both consider 3 candidate memories during utility-based selection, while MemRL uses a utility learning rate of 0.3 to update memory utility estimates. In addition, MemEff generates 3 candidate memories from each trajectory during memory induction. The LLM temperature is set to 0.8 for most methods, except AWM, which uses deterministic decoding with a temperature of 0. For a fair comparison, we follow the original implementations or papers for all baseline hyperparameters and use the same configurations across settings whenever applicable.

D.3 Evaluation Reliability

To assess the reliability of our evaluation and the stability of different memory optimization methods, we conduct five independent runs for each experimental setting and report the corresponding performance variance in Table 13. Compared with the mean performance reported in Table 1, these variance statistics characterize the consistency of each method across independent runs and provide

a complementary measure of evaluation reliability. The relatively small variances observed across the evaluated settings indicate that the reported performance is generally consistent across runs, supporting the robustness of our experimental evaluation.

Table 13: Variance of Performance over five independent runs. "Avg." denotes the average variance within each benchmark.

Inference Model	Method	WebArena-Lite						ALFWorld			ScienceWorld		
		Admin	GitLab	Map	Reddit	Shopping	Avg.	Seen	Unseen	Avg.	Seen	Unseen	Avg.
GPT-5.1	Static	2.45	0.00	0.67	1.59	2.03	1.35	0.44	1.69	1.06	0.59	0.54	0.57
	AWM	2.61	2.93	0.41	0.68	0.57	1.44	0.72	0.84	0.78	0.72	0.84	0.78
	Reasoning Bank	0.65	0.68	0.41	1.072	0.95	0.75	0.93	0.86	0.90	0.72	0.73	0.73
	CER	1.63	1.95	1.10	0.68	0.38	1.15	-	-	-	-	-	-
	JitRL	0.65	1.17	1.65	0.61	1.51	1.12	-	-	-	-	-	-
	MemRL	0.98	0.78	3.85	0.68	2.36	1.73	0.95	0.87	0.91	0.73	0.72	0.73
	MemEff (Our)	1.63	0.98	0.41	0.46	0.95	0.89	0.73	0.66	0.69	0.73	0.67	0.70
Gemma-4-31B-it	Static	3.02	0.85	0.00	2.95	0.38	1.44	0.73	0.69	0.71	0.77	1.17	0.97
	AWM	2.20	2.93	0.41	0.68	0.47	1.34	1.23	1.12	1.18	0.67	0.60	0.63
	Reasoning Bank	1.63	2.64	2.74	0.45	0.00	1.49	1.22	1.43	1.33	0.89	0.87	0.88
	CER	0.51	0.78	2.71	0.45	0.95	1.08	-	-	-	-	-	-
	JitRL	1.63	0.78	2.74	0.45	0.00	1.12	-	-	-	-	-	-
	MemRL	0.98	2.93	0.96	1.81	0.95	1.53	0.39	0.66	0.52	0.81	1.04	0.93
	MemEff (Our)	2.45	1.95	0.39	0.00	0.96	1.15	0.67	0.60	0.64	0.54	0.58	0.56

D.4 Implementation Details

In Section 4.3, we leverage the historical usage records H_i of candidate memories to predict their utility. Specifically, we employ a product kernel to jointly model the similarity between the current instruction and historical instructions, as well as the compatibility between their behavioral efficacy values. Since the behavioral efficacy of a candidate memory is estimated online and may fall outside the range observed in its usage history, we clip it to the empirical support of H_i:

$$\tilde{d}(\tau_q^i, c^i) = \mathrm{clip}\left(\tilde{d}(\tau_q^i, c^i), \min_{j \in H_i} d^j, \max_{j \in H_i} d^j\right), \tag{21}$$

where d^j denotes the behavioral efficacy of the j-th record in H_i. If the maximum predicted utility $\widehat{\Delta r}_{i^\star}(u) \geq 0$, the corresponding memory $c^{i^\star}$ is retrieved to guide the agent; otherwise, no memory is provided, and the agent executes the task under the memory-free policy.

In addition, to mitigate the prediction variance induced by cold-start in Figure 7, we skip utility estimation when $|H_i| \leq 3$ and fall back to the first-stage semantic retrieval. Equation 6 computes the log-probabilities of actions, where each action consists of multiple tokens. In practice, our method requires computing the agent's per-task performance $r_0(u)$, which can generally be obtained through multiple task executions in a memoryless setting or in the first iteration of memory optimization. These rollouts are also accounted for in Table 7.

We note that the behavioral efficacy values in our experiments are predominantly negative, including for memories extracted from successful trajectories. This is not contradictory: prepending a memory shifts the LLM's attention toward the abstract strategy described in the memory, which dilutes the probability mass assigned to the exact executed action tokens. Consequently, $d(\tau, c)$ measures the *magnitude of behavioral perturbation* induced by the memory rather than the absolute reinforcement of the executed actions. A memory that preserves a successful trajectory should therefore exhibit a small (close to zero) efficacy, whereas a memory that effectively diverts a failed trajectory should exhibit a large negative efficacy. This interpretation is consistent with the selection rule in Equation 7, which favors minimal perturbation for successes and maximal deviation for failures.

Note that Equation 1 explicitly conditions the association function f on $D(u, m)$ alone. This is motivated by two observations: (i) the same memory may induce different utility under different instructions, and (ii) our empirical analysis in subsection 4.1 reveals that f is non-monotonic in D, so its optimum cannot be located by greedily maximizing D. Consequently, the optimal retrieval policy in Equation 7 maximizes the *expected utility* $\mathbb{E}[f(D, u)]$ rather than the expected efficacy $\mathbb{E}[D]$ itself. Since f is unknown a priori, MemEff estimates it non-parametrically from the historical usage records $\mathcal{H}_i$ via kernel regression (Equation 12), and selects the memory with the highest predicted utility. This closes the gap between the formulation (Equation 7) and the algorithm: the kernel

estimator is a sample-based approximation of f, and the argmax in Equation 12 is the empirical counterpart of the argmax in Equation 7.

Prior work has demonstrated that self-critique can effectively improve the diversity of generated outputs Zhai et al. (2025); Liu et al. (2026). In Subsection 4.2, to generate diverse memory candidates from the same trajectory, we first use the standard memory induction prompt to generate an initial candidate. For subsequent candidates, we introduce the following prompts to encourage diverse abstractions of the same experience. For WebArena, we encourage alternative page groupings, reusable page usages, different sub-goal decompositions, and diverse skill abstractions. For ALFWorld and ScienceWorld, we encourage alternative search or exploration strategies, different task decompositions, explicit reasoning about state transitions and environmental conditions, and appropriate fallback decisions for successful trajectories, while analyzing failed ones from different failure perspectives. The prompts used to generate diverse memory candidates are provided below.

Prompt for Diverse Memory Generation in WebArena

```
Write another memory summary of this same web trajectory.  Do not
paraphrase the previous memory and keep the original output format.

Change the summary along these lines:
- Tie every page to a URL that appears in the trajectory.  Do not
invent URLs.
- Change the grouping.  Merge visits that are the same page with
different query parameters, or split views that do different jobs
(search results, item page, cart, form, list, settings).
- Change the usage you record.  State how a later task on this
website would use the page:  how to reach it, and what can be
searched, filtered, read, or submitted there.  Do not restate the
page title.
- Add a page the previous variant skipped if the goal depended on
it.  Drop a page that is only a transient click and has no reusable
content.
- Ignore pages already listed under "Existing summarized pages" to
avoid duplication.
```

Prompt for Diverse Memory Generation in ALFWorld and ScienceWorld

```
For successful trajectories:  Write a new script for this similar
task.  Do not paraphrase the previous scripts.  Generate a different
but valid strategy based on the successful trajectory.
Consider the following:
- Change the search or exploration strategy.  Consider alternative
locations, objects, or environmental states that may be relevant to
the task.
- Change the task decomposition.  Separate object discovery,
preparation, intermediate operations, and final completion when
appropriate.
- Explicitly capture necessary state changes or environmental
conditions, such as heating, cooling, cleaning, switching on an
object, or changing the state of an experimental setup.
- Include useful fallback decisions when an expected object,
location, state, or operation does not work as intended.
- Abstract concrete objects, locations, and experimental
entities into reusable types or relations rather than copying
trajectory-specific details or listing low-level actions.

For failed trajectories:  Write a new memory summary for this failed
trajectory.  Do not paraphrase the previous summaries.  Analyze
the trajectory from a different failure perspective and identify a
```

```
reusable strategy for avoiding the observed failure.

Consider the following:
- Identify the key decision, assumption, or state transition that
led to failure.
- Propose an alternative search or exploration strategy when the
original choice of location, object, or experimental condition was
ineffective.
- Identify missing or incorrect state changes, environmental
conditions, or operation sequences when relevant.
- Describe fallback decisions when an expected object, state,
observation, or operation is unavailable or produces an unexpected
result.
- Abstract the failure pattern into reusable guidance rather than
reproducing trajectory-specific actions.
```