

# MEMRL: SELF-EVOLVING AGENTS VIA RUNTIME REINFORCEMENT LEARNING ON EPISODIC MEMORY

Shengtao Zhang<sup>1,\*</sup>, Jiaqian Wang<sup>2,\*</sup>, Ruiwen Zhou<sup>3</sup>, Junwei Liao<sup>1,4</sup>, Yuchen Feng<sup>5</sup>, Weinan Zhang<sup>1,4</sup>, Ying Wen<sup>1,4</sup>, Zhiyu Li<sup>5</sup>, Feiyu Xiong<sup>5</sup>, Yutao Qi<sup>2</sup>, Bo Tang<sup>6,5,†</sup>, Muning Wen<sup>1,†</sup>

<sup>1</sup>Shanghai Jiao Tong University, <sup>2</sup>Xidian University, <sup>3</sup>National University of Singapore,

<sup>4</sup>Shanghai Innovation Institute, <sup>5</sup>MemTensor (Shanghai) Technology Co., Ltd.

<sup>6</sup>University of Science and Technology of China

## ABSTRACT

The hallmark of human intelligence is the ability to master new skills through Constructive Episodic Simulation—retrieving past experiences to synthesize solutions for novel tasks. While Large Language Models possess strong reasoning capabilities, they struggle to emulate this self-evolution: fine-tuning is computationally expensive and prone to catastrophic forgetting, while existing memory-based methods rely on passive semantic matching that often retrieves noise. To address these challenges, we propose MEMRL, a framework that enables agents to self-evolve via non-parametric reinforcement learning on episodic memory. MEMRL explicitly separates the stable reasoning of a frozen LLM from the plastic, evolving memory. Unlike traditional methods, MEMRL employs a Two-Phase Retrieval mechanism that filters candidates by semantic relevance and then selects them based on learned Q-values (utility). These utilities are continuously refined via environmental feedback in an trial-and-error manner, allowing the agent to distinguish high-value strategies from similar noise. Extensive experiments on HLE, BigCodeBench, ALFWorld, and Lifelong Agent Bench demonstrate that MEMRL significantly outperforms state-of-the-art baselines. Our analysis experiments confirm that MEMRL effectively reconciles the stability-plasticity dilemma, enabling continuous runtime improvement without weight updates.

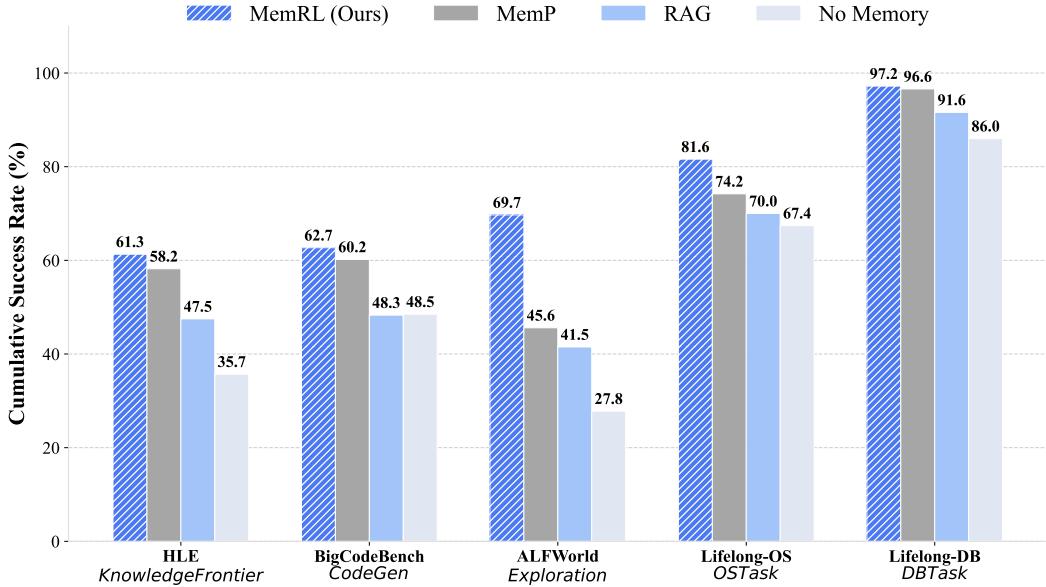


Figure 1: **Benchmark Runtime Learning performance of MEMRL.** We compare MEMRL against state-of-the-art memory baselines (MemP) and standard retrieval methods (RAG). **MEMRL** consistently outperforms various baselines, demonstrating the efficacy of runtime utility-driven updates.

\*Equal contribution. The order is decided by flipping a coin.

†Corresponding to Bo Tang (tangb@memtensor.cn) and Muning Wen (muningwen@sjtu.edu.cn)

## 1 INTRODUCTION

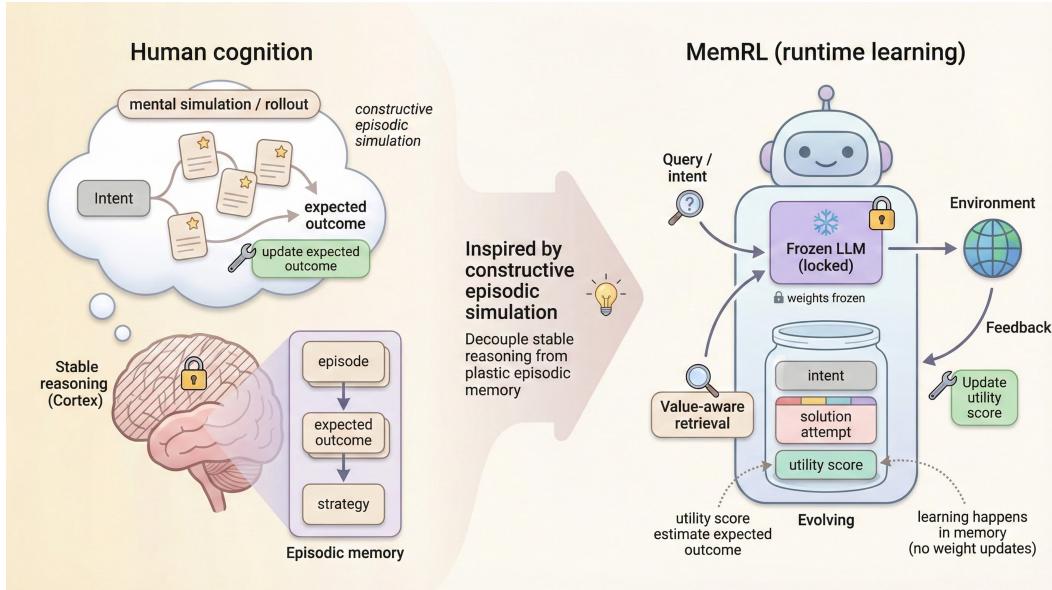


Figure 2: The conceptual framework of MEMRL.

The hallmark of human intelligence lies in the delicate balance between the stability of cognitive reasoning and the plasticity of episodic memory (Grossberg, 2013; McClelland et al., 1995; Kumaran et al., 2016), a mechanism known as Constructive Episodic Simulation that allows adaptation without rewiring neural circuitry (Schacter & Addis, 2007; Hassabis & Maguire, 2007; Schacter et al., 2012; Gick & Holyoak, 1980). While Large Language Models (LLMs) demonstrate impressive reasoning capabilities, existing paradigms struggle to emulate this dynamic, decoupled self-evolution (Wei et al., 2022; Yao et al., 2022; Schick et al., 2023; Wang et al., 2023). On one hand, fine-tuning approaches attempt to internalize experience by modifying model weights (Ouyang et al., 2022; Stiennon et al., 2020; Rafailov et al., 2023; Ethayarajh et al., 2024), but often suffer from *catastrophic forgetting* and high computational costs (Kirkpatrick et al., 2017; Li et al., 2024; Wu et al., 2024). On the other hand, Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) offers a non-parametric alternative but remains fundamentally passive; it retrieves information based solely on semantic similarity without evaluating its actual utility (Karpukhin et al., 2020; Gao et al., 2023). Lacking a mechanism to distinguish high-value past strategies from similar noise, current RAG agents struggle to effectively learn from runtime feedback to optimize their performance over time.

This limitation underscores a critical research question: **How can we enable an agent to continuously improve its performance after deployment, without compromising the stability of its pre-trained backbone?** Our objective is to achieve an agent that evolves with continued usage and rapidly adapts to new tasks after deployment, referred to as Runtime Continuous Learning (Javed et al., 2023; Silver & Sutton, 2025; Parisi et al., 2019; Wu et al., 2024), all while keeping the backbone model frozen to prevent catastrophic forgetting (Finn et al., 2017; Wei et al., 2025). To address this challenge, inspired by the human cognitive mechanism of constructive simulation, we propose **MEMRL**, a framework that facilitates self-evolving agents by explicitly decoupling the model’s stable cognitive reasoning from dynamic episodic memory. Figure 2 illustrates the conceptual framework of our proposed MEMRL. Drawing on value-iteration approaches in Reinforcement Learning (RL) to estimate expected experience utilities (Sutton & Barto, 2018), we formalize the interaction between the frozen LLM and external memory as a Markov Decision Process (MDP) (Puterman, 2014). Unlike traditional methods that optimize the backbone model, MEMRL optimizes the policy of memory usage to maximize expected utility.

MEMRL organizes memory into a structured **Intent-Experience-Utility** triplet. This structure transforms retrieval from a passive semantic match task into an active decision-making process: **Value-Aware Retrieval** selects experiences based on their learned Q-values, reflecting expected

utility, rather than semantic similarity alone (Watkins & Dayan, 1992); **Utility-Driven Update** refines these Q-values via environmental feedback and Bellman backup (Bellman, 1966). This closed-loop cycle enables the agent to distinguish high-value strategies from similar noise, effectively learning from both success and failure without the computational cost or catastrophic forgetting risks associated with weight updates. As for experiments, we validate MEMRL on four diverse benchmarks, including HLE, BigCodeBench, ALFWorld, and Lifelong Agent Bench. Our results demonstrate consistent superiority over baselines, achieving relative improvement in exploration-heavy environments. Our in-depth analysis reveals a strong correlation between learned utility and task success, further confirming MEMRL’s effectiveness.

In summary, our contributions are threefold:

- We propose a runtime learning framework based on Model-Memory decoupling and the Intent-Experience-Utility triplet, which reconciles the stability-plasticity dilemma by enabling agents to learn without parameter updates.
- We introduce MEMRL, a non-parametric reinforcement learning algorithm that implements Value-Aware Retrieval and Utility-Driven Memory Curation. This allows agents to self-evolve by optimizing memory utility, establishing a new paradigm for enhancing agent capabilities.
- We conduct extensive evaluations and provide deep insights into MEMRL’s working mechanism. We analyze how it ensures structural integrity in complex tasks and theoretically substantiate its stability via Bellman contraction, exploring how utility-driven updates minimize catastrophic forgetting while maximizing positive transfer.

## 2 RELATED WORKS

### 2.1 CONTINUOUS LEARNING

Continual learning addresses the stability-plasticity dilemma, aiming to acquire new knowledge sequentially without suffering from catastrophic forgetting. Classical approaches—such as regularization, distillation, and experience replay—mitigate forgetting by constraining parameter updates or preserving past data distributions (Kirkpatrick et al., 2017; Li & Hoiem, 2017; Lopez-Paz & Ranzato, 2017). However, these parametric methods are computationally expensive for LLMs and risk destabilizing the pre-trained backbone through frequent online updates. Recent surveys on continual learning for LLMs further systematize these difficulties and emphasize the importance of external mechanisms and non-parametric pathways (Wu et al., 2024). Therefore, from a continual learning perspective, if we aim for agents to improve with use while preserving the stability of the backbone, a more practical direction is to shift plasticity from the parameter space to external structures and controlled experience-update channels.

### 2.2 REINFORCEMENT LEARNING

Reinforcement learning has been widely adopted for LLMs enhancement. A representative paradigm is to construct reward signals from human feedback and optimize the model policy accordingly to align with human preference (Stienon et al., 2020; Ouyang et al., 2022). Other recent approaches leverage rule-based verifiers to improve LLMs’ reasoning capabilities (Guo et al., 2025; Yu et al., 2025). In parallel, agent-oriented research explores how interaction signals can improve tool use and action decision-making, and investigates mechanisms by which language models execute composite actions in environments (Schick et al., 2023). Despite the demonstrated effectiveness of reward-driven optimization, these methods generally place learning in the model parameters or additional parametric modules, and thus do not fundamentally avoid the cost of online updates or the risk of forgetting. In contrast, our method frames memory usage as a learnable decision problem and applies *non-parametric* reinforcement learning on memory to bypass the risk.

### 2.3 AGENTIC MEMORY

To bypass the costs of fine-tuning, external memory systems have evolved from a static RAG paradigm to dynamic, governable memory structures (Lewis et al., 2020; Karpukhin et al., 2020). Early agentic memory introduced reflection mechanisms and hierarchical management to handle long context

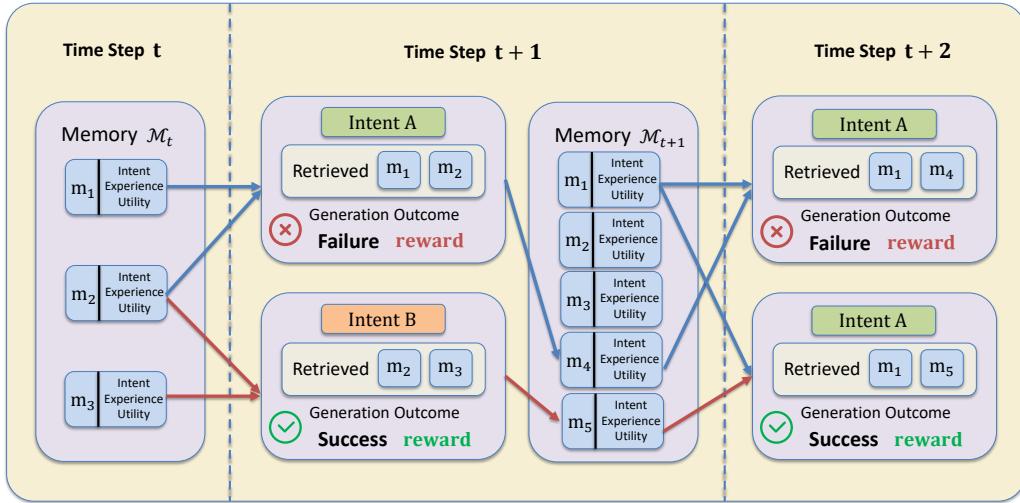


Figure 3: An illustrative example of memory-augmented decision making under a Markov Decision Process. At time step  $t$ , the agent starts with an initial memory set  $\mathcal{M}_t$ . At time step  $t+1$ , an intent (*Intent A*) retrieves relevant past experiences, but initially leads to a failed generation. In contrast, another intent (*Intent B*) succeeds and its associated experience is added to memory. At time step  $t+2$ , *Intent A* retrieves the newly stored successful experience from *Intent B*, resulting in a successful outcome. This example shows how memory retrieval enables knowledge reuse across intents, implicitly supporting transfer across tasks through shared experiences.

experiences (Shinn et al., 2023; Packer et al., 2024). More recent frameworks have systematized the memory lifecycle, focusing on unified storage and structured indexing for complex tasks (Li et al., 2025b; Xu et al., 2025; Huang et al., 2025; Ye, 2025). Furthermore, adaptive approaches now explore improving retrieval via feedback-driven updates or automated augmentation (Salama et al., 2025; Zhang et al., 2025; Li et al., 2025a; Zhou et al., 2025). However, except for training additional learnable modules, most existing methods still rely predominantly on semantic similarity or heuristic rules, lacking a rigorous metric to evaluate the actual utility of a memory in maximizing returns. Inspired by cognitive theories of memory reconsolidation (Schacter & Addis, 2007; Gick & Holyoak, 1980; Nader et al., 2000), MEMRL bridges this gap by formulating retrieval as a value-based decision process, learning robust utility estimates (Q-values) directly from environmental rewards to distinguish high-value experiences from noise.

### 3 PROBLEM FORMULATION

In this section, we formally define the problem of memory-augmented generation and establish the theoretical link between agent policy and memory retrieval. We adopt the formulation of Memory-Based Markov Decision Process (M-MDP) (Zhou et al., 2025) and address it with our non-parametric reinforcement learning framework. Figure 3 provides an illustrative example of this memory-augmented decision process, showing how retrieval outcomes and memory evolution unfold over multiple time steps.

#### 3.1 MEMORY-AUGMENTED AGENT POLICY

To enable the agent to self-evolve, we inherit the M-MDP as the problem formulation (Zhou et al., 2025). The M-MDP is formally defined as a tuple  $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mathcal{M} \rangle$ , where  $\mathcal{S}$  denotes the state space,  $\mathcal{A}$  the action space,  $\mathcal{P} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$  the transition dynamics,  $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$  the reward function,  $\gamma \in [0, 1]$  the discount factor, and  $\mathcal{M} = (\mathcal{S} \times \mathcal{A} \times \mathbb{R})^*$  the evolving memory space containing past experiences (Zhou et al., 2025). In this setting, the policy  $\pi$  not only generates tokens

but first selects a memory context  $m$  based on a retrieval distribution  $\mu(\cdot|s, \mathcal{M})$ , allowing the agent to leverage historical data for better performance in downstream tasks.

We consider a generalist agent interacting with an environment or user over discrete time steps. At each step  $t$ , the agent receives a state  $s_t$ , e.g., a user query or task description, and has access to an external memory bank  $\mathcal{M}$ . The agent’s objective is to generate a response  $y_t$  that maximizes a reward signal. Following this formulation, the behavior of a memory-augmented agent can be decomposed into two distinct phases: *Retrieve* and *Generation*. The joint policy  $\pi(y_t|s_t, \mathcal{M}_t)$  is defined as the marginal probability over all possible retrieved memory items (Zhou et al., 2025):

$$\pi(y_t|s_t, \mathcal{M}_t) = \sum_{m \in \mathcal{M}_t} \underbrace{\mu(m|s_t, \mathcal{M}_t)}_{\text{Retrieval Policy}} \cdot \underbrace{p_{\text{LLM}}(y_t|s_t, m)}_{\text{Inference Policy}} \quad (1)$$

where:

- $\mu(m|s_t, \mathcal{M}_t)$  represents the **Retrieval Policy**, which assigns a probability to selecting a specific memory context  $m$  composed of past intents and experiences. from the memory bank  $\mathcal{M}_t$  given the current state  $s_t$ .
- $p_{\text{LLM}}(y_t|s_t, m)$  represents the **Inference Policy**, typically parameterized by a frozen LLM. It models the likelihood of generating output  $y_t$  conditioned on both the query  $s_t$  and the retrieved context  $m$ .

In previous RAG or memory-based agentic paradigms, the retrieval policy  $\mu$  is usually determined by a fixed vector similarity metric, e.g., cosine similarity of embeddings. While effective for semantic matching, such policies fail to account for the *utility* of a memory, i.e., whether retrieving  $m$  actually leads to a successful outcome  $y_t$ .

### 3.2 NON-PARAMETRIC REINFORCEMENT LEARNING

To overcome the limitations of static similarity-based retrieval, we operationalize the M-MDP framework by formulating memory retrieval as a value-based decision-making process. Unlike parametric approaches that optimize  $\pi_{\text{LLM}}$  via weight updates, we aim to optimize the retrieval policy  $\mu(m|s, \mathcal{M})$  directly within the memory space. We map the abstract M-MDP components to a specific Intent-Experience-Utility structure:

**From Semantic Matching to Decision Making.** We instantiate the state  $s$  as the User Intent, encapsulated by the embedding of the current query (Lewis et al., 2020). Consequently, the action space  $\mathcal{A}_t$  becomes dynamic and discrete, corresponding to the selection of a specific  $m$  from the current memory bank  $\mathcal{M}_t$ . Under this formulation, retrieving from memory is no longer a passive matching task but an active action  $a_t = m$  taken to augment the generator (Zhou et al., 2025).

**Defining Utility via Q-Values.** The core of our framework is the shift from estimating semantic relevance to estimating functional utility. We define the state-action value function  $Q(s, m)$  as the expected utility of applying the retrieved context  $m$  for intents similar to  $s$ . The objective of MEMRL is to learn an optimal retrieval policy  $\mu^*$  that selects context maximizing this expected utility:

$$\mu^*(m|s, \mathcal{M}) = \arg \max_{m \in \mathcal{M}} Q(s, m) \quad (2)$$

This  $Q$ -value serves as a critic, distinguishing high-value strategies from irrelevant noise that may share high semantic similarity.

**Non-Parametric Learning.** Since the action space of the retrieval policy  $\mu$  in our setting is decoupled from the LLMs’ generation space, we can perform learning without modifying the LLMs’ weights. Upon receiving an environmental feedback  $r$ , e.g., execution success, we can update the Q-value of the retrieved memory context using a Temporal-Difference (TD) error (Sutton, 1988):

$$Q(s, m) \leftarrow Q(s, m) + \alpha[r + \gamma \max Q(s', m') - Q(s, m)] \quad (3)$$

where  $\alpha$  is the learning rate. This Bellman-style backup allows the utility estimates to converge to the true expected returns over time (Bellman, 1966). By maintaining and updating these Q-values explicitly within the memory structure, MEMRL provides a non-parametric learning manner with a theoretical guarantee that enables the agent to self-evolve its capabilities through interaction.

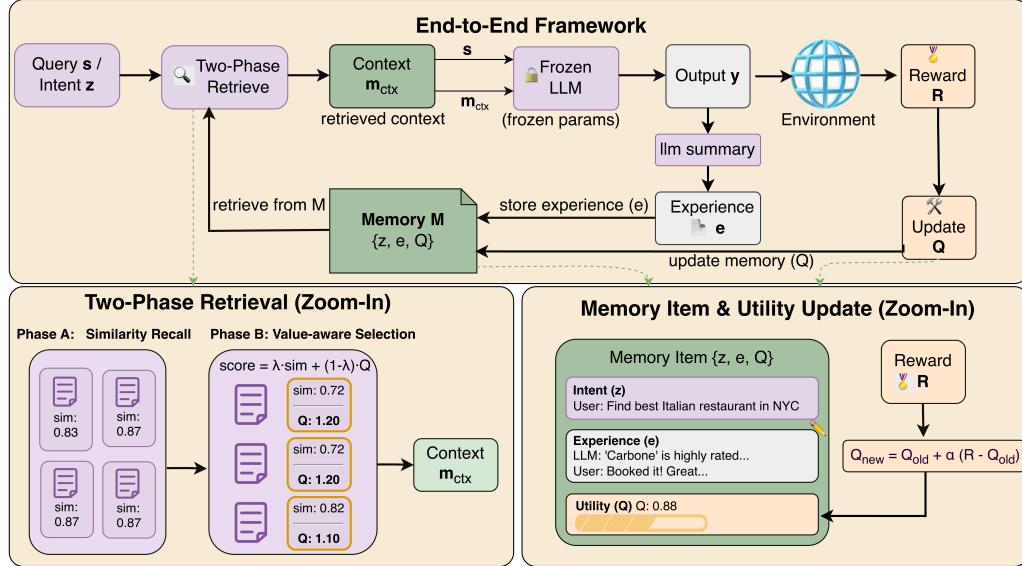


Figure 4: **Overview of the MEMRL Framework.** **(Top)** The end-to-end learning loop: given a query  $s$ , the agent retrieves context  $m_{ctx}$  from memory  $M$ , generates output  $y$ , and updates memory value  $Q$  based on reward  $R$ . **(Bottom Left)** Two-Phase Retrieval: Candidates are recalled via similarity, then re-ranked using learned  $Q$ -values. **(Bottom Right)** Utility Update: Memory values ( $Q$ ) are updated using environmental rewards to distinguish functional utility from semantic similarity.

## 4 MEMRL

Building upon the M-MDP formulation defined in Sec. 3, we propose MEMRL, a framework that enables frozen LLMs to self-evolve via non-parametric reinforcement learning. Instead of modifying the model weights  $\theta$ , MEMRL optimizes the **retrieval policy**  $\mu(m|s, \mathcal{M})$  within an evolving memory space. As illustrated in Figure 4, the framework consists of three core components: (i) a structured **Intent-Experience-Utility** memory bank, (ii) a **Two-Phase Retrieval** mechanism that decouples semantic recall from value-aware selection, and (iii) a **Runtime Utility Update** rule that stabilizes  $Q$ -value estimation.

### 4.1 MEMORY STRUCTURE: THE INTENT-EXPERIENCE-UTILITY TRIPLET

To support value-based decision-making, we structure the external memory  $\mathcal{M}$  not merely as key-value pairs, but as a set of triplets:

$$\mathcal{M} = \{(z_i, e_i, Q_i)\}_{i=1}^{|\mathcal{M}|}, \quad (4)$$

where  $z_i$  represents the **Intent Embedding** (e.g., the vector representation of a query or task description),  $e_i$  stores the raw **Experience** (e.g., a successful solution trace or trajectory), and  $Q_i \equiv Q(z_i, e_i)$  denotes the learned **Utility**.  $Q_i$  approximates the expected return of applying experience  $e_i$  to intents similar to  $z_i$ , serving as the *critic* in RL formulation.

### 4.2 TWO-PHASE RETRIEVAL: FROM SEMANTIC RECALL TO VALUE-AWARE SELECTION

Standard RAG or memory systems rely solely on semantic similarity, implicitly assuming that “similar implies useful.” However, in agentic tasks, semantically relevant contexts may encode brittle, environment-specific routines that fail to generalize (Singh et al., 2025; Cuconasu et al., 2024; Gan et al., 2024; Zhou et al., 2024). To address this, MEMRL implements a Two-Phase Retrieval strategy that filters candidates first by relevance, then by utility.

**Phase A: Similarity-Based Recall.** Given a current query state  $s$ , we first isolate a candidate pool  $\mathcal{C}(s)$  of semantically consistent experiences to ensure the retrieval is contextually relevant. We

compute the cosine similarity  $\text{sim}(s, z_i)$  and filter the memory bank:

$$\mathcal{C}(s) = \text{TopK}_{k_1}(\{i | \text{sim}(s, z_i) > \delta\}, \text{by sim}) \quad (5)$$

where  $\delta$  is a sparsity threshold. This phase acts as a coarse filter, reducing the search space from the entire memory  $\mathcal{M}$  to a relevant subset  $\mathcal{C}(s)$ . Notably, if  $\mathcal{C}(s) = \emptyset$ , MEMRL injects no memory and relies solely on the frozen LLM for broader exploration.

**Phase B: Value-Aware Selection.** To select the optimal context from  $\mathcal{C}(s)$ , we incorporate the learned utility  $Q$ . We define a composite scoring function that balances exploration (via semantic matching) and exploitation (via high-utility history):

$$\text{score}(s, z_i, e_i) = (1 - \lambda) \cdot \hat{\text{sim}}(s, z_i) + \lambda \cdot \hat{Q}(z_i, e_i) \quad (6)$$

where  $\hat{\cdot}$  denotes z-score normalization within the candidate pool, and  $\lambda \in [0, 1]$  modulates the trade-off. As  $\lambda \rightarrow 1$ , the policy prioritizes proven utility; as  $\lambda \rightarrow 0$ , it reverts to standard similarity-based retrieval. The final context  $\mathcal{M}_{ctx}(s)$  consists of the top- $k_2$  items maximizing this score:

$$\mathcal{M}_{ctx}(s) = \text{TopK}_{k_2}(\mathcal{C}(s), \text{by score}). \quad (7)$$

This mechanism effectively filters out “distractor” memories—those that are semantically similar but historically yielded low returns (low Q-values). We further validate the necessity of normalization and similarity threshold in Section 5.3.3, demonstrating that z-score normalization and strict similarity threshold are essential for filtering noise and maintaining low forgetting rates during self-evolution.

#### 4.3 RUNTIME LEARNING: NON-PARAMETRIC RL ON MEMORY

The core of MEMRL is the continuous refinement of Q-values based on environmental feedback, enabling the agent to “remember” what works. During runtime, MEMRL performs learning entirely in memory space. After completing a task, the agent receives an environmental reward signal  $r$  (e.g., execution success, user feedback, or scalar task score). For the memories actually injected into the context  $\mathcal{M}_{ctx}(s)$ , we update their utilities in triplets with a Monte Carlo style rule (Metropolis & Ulam, 1949):

$$Q_{\text{new}} \leftarrow Q_{\text{old}} + \alpha(r - Q_{\text{old}}). \quad (8)$$

Equation 8 performs as a naturally simplified version of Equation 3 by setting  $s'$  as a terminal state, sharing a similar one-step MDP formulation with Guo et al. (2025). This update drives  $Q_{\text{new}}$  toward the empirical expected return of using experience  $e_i$  under similar intents. Meanwhile, for each sampled trajectory, we use an LLM to summarize the experience, and write it back into the memory bank as a new triplet  $(z(s), e_{\text{new}}, Q_{\text{init}})$ , enabling continual expansion of experience while keeping the LLM parameters unchanged.

#### 4.4 COGNITIVE INTERPRETATION

MEMRL provides an algorithmic analogue of constructive episodic simulation (Schacter & Addis, 2007). Phase-A operationalizes **analogical transfer** (Gick & Holyoak, 1983) by recalling semantically similar past events. Phase-B resembles **mental rehearsal** (Cisek & Kalaska, 2004) by selecting among recalled candidates using learned utility estimates, effectively favoring strategies that expectably led to higher returns. Finally, Eq. 8 implements a form of **memory reconsolidation** (Haubrich & Nader, 2016): once a memory is retrieved and applied, its utility is reinforced or attenuated according to subsequent outcomes. Together, these components realize a stability–plasticity balance: the frozen LLM preserves stable cognitive reasoning, while the evolving memory utilities provide the plastic channel for continual adaptation.

#### 4.5 STABILITY ANALYSIS

We analyze the stability of MEMRL from a reinforcement learning perspective, focusing on the convergence behavior of the utility estimates stored in memory. Unlike classical value iteration, MEMRL performs non-parametric runtime learning using a constant-step-size update. We show that under mild and realistic assumptions, the learned utility values converge in expectation to stable estimates of memory effectiveness, with bounded variance.

**Setup.** At each time step  $t$ , the agent observes an intent state  $s_t$ , retrieves a memory item  $m_t \in \mathcal{M}_t$ , generates an output  $y_t$ , and receives a scalar reward  $r_t \in [-1, 1]$  indicating task success or failure. The generation policy follows the decomposition defined in Eq. 1, where  $\mu$  denotes the retrieval policy and  $p_{\text{LLM}}$  is a frozen inference policy.

For each retrieved memory, MEMRL updates its utility using the exponential moving average rule as formulated in Eq. 8, with learning rate  $\alpha \in (0, 1]$ . For clarity in this analysis, we consider a fixed state–memory pair  $(s, m)$  and write  $Q_t \equiv Q_t(s, m)$ .

**Stationary Reward Assumption.** We analyze the learning process on a fixed dataset and posit two key conditions that ensure the stability of the environment:

1. **Frozen Inference Policy.** The parameters of  $p_{\text{LLM}}(y|s, m)$  and the evaluator’s criteria are fixed.
2. **Fixed Task Distribution.** Tasks  $s$  are drawn from a stationary distribution over a fixed dataset.

These assumptions guarantee that the learning target is well-defined: the expected reward for any specific task–memory pair is time-invariant. Thus, we have:

$$\mathbb{E}[r_t|s_t = s, m_t = m] = \beta(s, m). \quad (9)$$

**Expected Convergence of Utility Estimates.** We now state the main stability result.

**Theorem 1.** Let  $\{Q_t\}$  be updated according to the rule in Eq. 8 with constant step size  $\alpha \in (0, 1]$ . If Eq. 9 holds and the pair  $(s, m)$  is updated infinitely often, then:

$$\lim_{t \rightarrow \infty} \mathbb{E}[Q_t] = \mathbb{E}[r_t|s_t = s, m_t = m] = \beta(s, m). \quad (10)$$

Moreover, the convergence rate is exponential:

$$\mathbb{E}[Q_t] - \beta(s, m) = (1 - \alpha)^t (Q_0 - \beta(s, m)). \quad (11)$$

**Proof.** Define the estimation error  $e_t \triangleq Q_t - \beta(s, m)$ . Based on the update rule in Eq. 8, the error recurrence relation is:

$$e_{t+1} = (1 - \alpha)e_t + \alpha(r_t - \beta(s, m)).$$

Taking conditional expectation given the history  $\mathcal{F}_t$  and using Eq. 9, we obtain:

$$\mathbb{E}[e_{t+1}|\mathcal{F}_t] = (1 - \alpha)e_t.$$

Taking full expectation yields:

$$\mathbb{E}[e_{t+1}] = (1 - \alpha)\mathbb{E}[e_t].$$

Iterating the recursion gives  $\mathbb{E}[e_t] = (1 - \alpha)^t e_0$ , which converges to zero as  $t \rightarrow \infty$ .  $\square$

We provide the detailed derivation of the convergence proof in Appendix A.1.

**Bounded Variance and Stability.** If the reward variance  $\text{Var}(r_t|s, m) < \infty$ , then the variance of  $Q_t$  remains bounded:

$$\limsup_{t \rightarrow \infty} \text{Var}(Q_t) \leq \frac{\alpha}{2 - \alpha} \text{Var}(r_t|s, m). \quad (12)$$

Thus, constant-step-size updates do not induce unbounded oscillations; instead, they yield stable utility estimates that track expected memory effectiveness while filtering high-frequency noise. We explicitly derive the variance bounds to demonstrate the global stability of the estimator under task clustering in Appendix A.2.

**Global Stability via EM Convergence.** The stability of the local estimate (Theorem 1) extends to the global memory utility  $Q(m)$ . By the linearity of expectation,  $Q(m)$  acts as a Monte Carlo integrator striving to converge to:

$$\lim_{t \rightarrow \infty} \mathbb{E}[Q_t(m)] = \mathbb{E}[r|m] = \sum_{s \in \mathcal{S}(m)} \underbrace{\mathbb{E}[r|s, m]}_{\text{Stationary}} \underbrace{\Pr(s|m)}_{\text{Retrieve-Dependent}}. \quad (13)$$

Table 1: **Runtime Learning main results.** We compare MEMRL against various baselines over 10 epochs. The results are reported as **Last Epoch Accuracy / Cumulative Success Rate (CSR)**. CSR indicates the percentage of tasks solved at least once during the training process. “–” indicates the experiment was not applicable.

<b>Method</b>	<b>BigCodeBench</b>		<b>Lifelong Agent Bench</b>		<b>ALFWorld</b>	<b>HLE</b>
	Code Gen (Last / CSR)	OS Task (Last / CSR)	DB Task (Last / CSR)	Exploration (Last / CSR)	Knowledge Frontier (Last / CSR)	
<i>Model</i>	GPT-4o	GPT-4o-mini	GPT-4o-mini	GPT-4o-mini	GPT-4o-mini	Gemini-3-pro
No Memory	0.485	0.674	0.860	0.278	0.357	
Pass@10	– / 0.577	– / 0.756	– / 0.928	– / 0.462	– / 0.524	
Reflexion	– / 0.614	– / 0.714	– / 0.938	– / 0.358	– / –	
RAG	0.475 / 0.483	0.690 / 0.700	0.914 / 0.916	0.370 / 0.415 <sup>†</sup>	0.430 / 0.475	
Self-RAG	0.497 / 0.561	0.646 / 0.732	0.891 / 0.898	0.290 / 0.290 <sup>†</sup>	0.411 / 0.427 <sup>†</sup>	
Mem0	0.487 / 0.495	0.670 / 0.702	0.920 / 0.926	– / –	0.400 / 0.406 <sup>†</sup>	
MemP	0.578 / 0.602	0.736 / 0.742	<b>0.960</b> / 0.966	0.324 / 0.456	0.528 / 0.582	
<i>MemRL (ours)</i>	<b>0.595 / 0.627</b>	<b>0.794 / 0.816</b>	<b>0.960 / 0.972</b>	<b>0.507 / 0.697</b>	<b>0.573 / 0.613</b>	

<sup>†</sup> Experiments are still running, results are reported using fewer than 10 training epochs.

where  $\mathcal{S}(m) \triangleq \{s \in \mathcal{S} | \text{sim}(s, z_m) \geq \tau_A\}$  denotes the *effective support set* for memory  $m$ , comprising all task intents  $s$  sufficiently similar to the memory’s intent embedding  $z_m$  to satisfy the Phase-A retrieval criterion.

A theoretical challenge arises here: the weighting term  $\Pr(s|m)$  is a latent variable governed by the retrieval policy  $\mu(m|s; \mathcal{M})$ , which itself shifts as  $Q$ -values evolve. To prove convergence despite this dependency, we analyze MEMRL as a **Generalized Expectation-Maximization (GEM)** process (Dempster et al., 1977; Neal & Hinton, 1998). From a variational perspective, the system performs coordinate ascent on a global objective function  $\mathcal{J}(Q, \mu)$  (the variational lower bound of expected reward): (i) **E-Step (Policy Improvement):** The Phase-B ranking updates the retrieval policy  $\mu$  to align with current estimates, monotonically increasing  $\mathcal{J}$  with respect to  $\mu$ ; (ii) **M-Step (Value Update):** The utility update (Eq. 8) increases  $\mathcal{J}$  with respect to  $Q$ . By the *Monotonic Improvement Theorem* (Neal & Hinton, 1998), this alternating optimization guarantees that the system converges to a stationary point where the retrieve policy stabilizes ( $\mu_{t+1} \approx \mu_t$ ). Consequently, the induced distribution  $\Pr(s|m)$  becomes time-invariant, ensuring that Eq. 13 holds and effectively preventing catastrophic forgetting by anchoring updates to a stable policy. More details can be found in Appendix B.

## 5 EXPERIMENTS

### 5.1 EXPERIMENTAL SETUP

**Baselines.** We evaluate MEMRL against a comprehensive suite of memory-augmented baselines under a **unified frozen-backbone setting** to isolate the contribution of memory mechanisms: (i) **RAG-based Approaches:** *RAG* (Lewis et al., 2020) and *Self-RAG* (Asai et al., 2023), representing standard semantic retrieval and critique-based filtering, respectively. (ii) **Agentic Memory:** *Mem0* (Chhikara et al., 2025) and *MemP* (Fang et al., 2025), which introduce structured write/read operations or procedural memory distillation. (iii) **Test-Time Scaling:** *Pass@k* and *Reflexion*, which applies iterative self-refinement over  $k$  rounds (Shinn et al., 2023).

**Benchmarks.** We evaluate MEMRL and baselines on four diverse benchmarks: **BigCodeBench** (Zhuo et al., 2025) for code generation, **ALFWorld** (Shridhar et al., 2021) for embodied navigation, **Lifelong Agent Bench** (Zheng et al., 2025) for OS/DB interaction, and **Humanity’s Last Exam (HLE)** (Phan et al., 2025) for multidisciplinary complex reasoning.

We evaluate our MEMRL and baselines under two distinct settings: **Runtime Learning**, which assesses the ability to learn and adapt within a training session, and **Transferring**, which evaluates the generalization capability of the learned memory on unseen tasks.

Table 2: **Transfer Learning results** on BigCodeBench, Lifelong Agent Bench and ALFWorld. We compare our MEMRL against various retrieval and memory baselines using best validation results.

<b>Method</b>	<b>BigCodeBench</b>	<b>Lifelong Agent Bench</b>		<b>ALFWorld</b>
	Code Generation (Acc)	OS Task (Acc)	DB Task (Acc)	Exploration (Acc)
<i>Model</i>	GPT-4o	GPT-4o-mini	GPT-4o-mini	GPT-4o-mini
No Memory	0.485	0.673	0.841	0.314
RAG	0.479	0.713	0.920	0.336 <sup>†</sup>
Self-RAG	0.500	0.653	0.881	0.293 <sup>†</sup>
Mem0	0.485	0.686	0.935	—
MemP	0.494	0.720	0.928	0.421
<i>MemRL (ours)</i>	<b>0.508</b>	<b>0.746</b>	<b>0.942</b>	<b>0.479</b>

<sup>†</sup> Experiments are still running, results are reported using fewer than 10 training epochs.

**Runtime Learning Results.** Table 1 reports the Last Epoch Accuracy (Last) and Cumulative Success Rate (CSR) over 10 training epochs. MEMRL consistently outperforms all baselines across all domains, validating that non-parametric value estimation effectively guides self-evolution. Notably, the advantages of our value-aware retrieval are most pronounced in exploration-heavy environments such as ALFWorld. Here, MEMRL achieves a remarkable last-epoch accuracy of 0.507, representing a relative improvement of approximately 56% over MemP (0.324) and 82% over the No Memory baseline (0.278). Furthermore, the high CSR of 0.697 in ALFWorld indicates that the RL component effectively encourages the agent to explore and discover solutions for complex tasks that similarity-based retrieval methods often fail to solve. Similarly, in the challenging Knowledge Frontier HLE benchmark, MEMRL improves the last accuracy to 0.573 compared to 0.528 for MemP, where the CSR even reached an astonishing 61.3%. Comparing these gains against single-turn tasks like BigCodeBench reveals an important trend: the performance uplift correlates with task structural complexity. The benefits of MEMRL are maximized in environments characterized by deep exploration and high procedural transferability (e.g., ALFWorld), whereas the margin is narrower in tasks with lower structural reuse. This suggests that our value-based mechanism is particularly adept at distilling and transferring complex problem-solving patterns from exploratory trajectories.

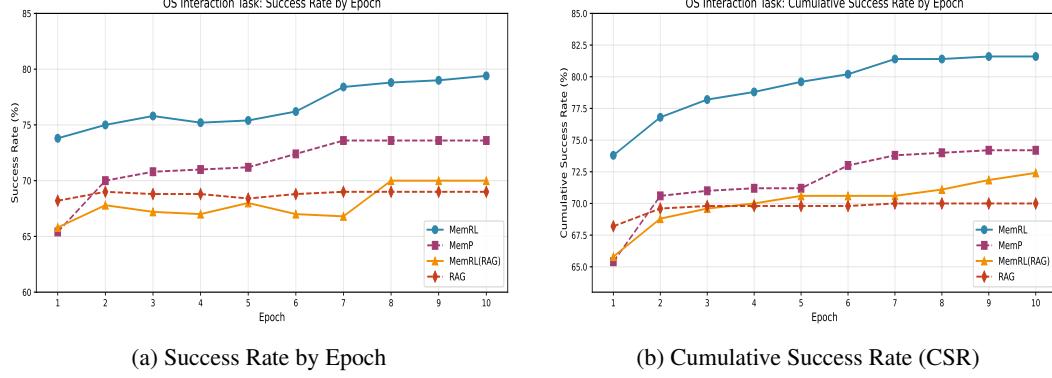
Overall, the simultaneous improvement in both CSR and Last Accuracy suggests that MEMRL not only discovers high-quality solutions during exploration but also effectively retains and retrieves them for stable performance.

**Transferring Results.** We evaluate memory transferability by freezing the memory bank after training and testing on held-out sets (30% split). As shown in Table 2, MEMRL exhibits superior generalization compared to baseline methods. On BigCodeBench, MEMRL achieves the highest accuracy of 0.508, outperforming advanced retrieval methods such as Self-RAG (0.500) and standard MemP (0.494). In the OS control tasks (Lifelong Agent Bench), our method attains an accuracy of 0.746, significantly improving upon the standard RAG baseline (0.713). Consistent with the runtime results, the gain is substantial in ALFWorld, where MEMRL reaches 0.479, demonstrating a clear margin over MemP (0.421) and RAG (0.336). These results validate that the **Value-Aware Retrieval** and **Non-Parametric RL** mechanism in MEMRL does not merely overfit to training instances; instead, it filters out low-value memories, retaining high-utility experiences that facilitate generalization to unseen tasks.

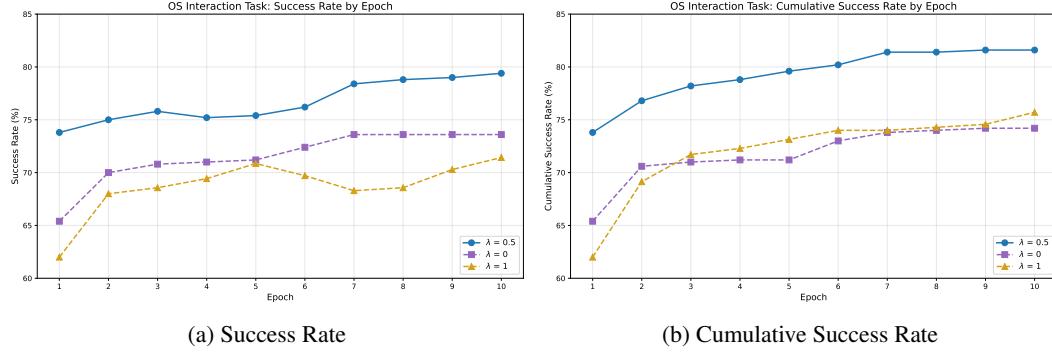
## 5.2 ABLATIONS

### 5.2.1 EFFECTIVENESS OF RUNTIME RL

To isolate the effect of runtime RL, we compare the memory mechanisms with and without the RL component (i.e., MemP vs. MEMRL and RAG vs. MEMRL(RAG)) within the OS interaction environment. Figure 5a illustrates the success rate evolution across epochs. While the RL enhancement yields marginal gains in the initial phase, a clear performance divergence emerges as training progresses.



**Figure 5: OS Interaction Performance.** (a) MEMRL demonstrates superior stability and higher peak performance compared to the baseline. (b) The widening gap in CSR illustrates that RL-enhanced methods effectively accumulate solved tasks over time.



**Figure 6: Ablation study on Q-value weighting factor  $\lambda$ .** The balanced setting ( $\lambda = 0.5$ ) achieves superior stability and peak performance compared to extreme configurations.

MEMRL consistently outperforms the vanilla MemP baseline in later epochs, exhibiting a smoother learning curve with fewer regressions. This stability suggests that the RL-driven value function effectively mitigates the interference of noisy memories that often plague static similarity-based retrieval.

We further analyze the CSR in Figure 5(b), which measures the agent’s ability to solve distinct tasks at least once during the training process. Here, the benefits of RL are even more pronounced. MEMRL and MEMRL(RAG) attain final CSR outperforming their counterparts MemP and RAG, respectively. The monotonic widening of the performance gap in Figure 5(a) indicates that runtime RL significantly enhances the agent’s ability to recover from early failures. By prioritizing memories with high expected utility, the agent effectively mitigates the interference of noisy memories and consolidates successful experiences, transforming transient exploration into robust, repeatable capabilities.

### 5.2.2 IMPACT OF Q-VALUE WEIGHTING

To understand the interplay between semantic retrieval and reinforcement learning, we analyze the impact of the Q-weighting factor  $\lambda$  in the scoring function, i.e., the Equation 6: We compare the balanced setting ( $\lambda = 0.5$ ) against two extremes: pure semantic retrieval ( $\lambda = 0$ ) and pure greedy RL ( $\lambda = 1$ ).

As illustrated in Figure 6, the **balanced configuration** consistently yields superior performance. This indicates that the combination of semantic grounding and utility-driven ranking effectively guides the agent towards high-quality solutions while maintaining context relevance. Besides, the **pure semantic retrieval baseline** provides a stable starting point but lacks the mechanism to filter out suboptimal memories that are semantically similar but functionally incorrect. Consequently, its performance plateaus early, which highlights the necessity of the RL signal for continued improvement. On the other hand, the **pure RL setting** reveals the risks of disregarding semantic context, exhibiting

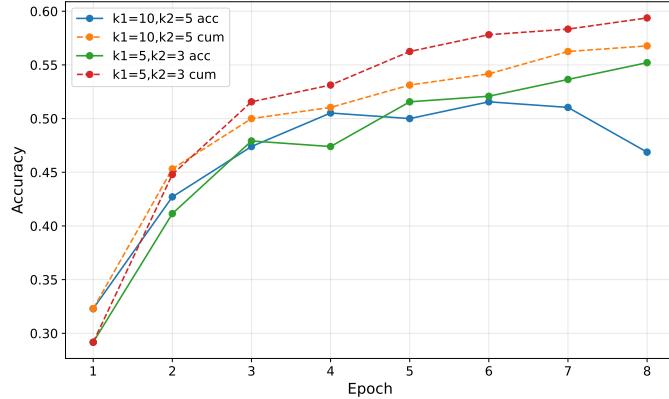


Figure 7: **Ablation on Retrieval Size ( $k_1, k_2$ ).** Performance (acc: Epoch-Acc; cum: Cumulative Acc) comparison on HLE (CS/AI) across different retrieval counts.

significant instability and poor initial performance. We attribute this instability to *context detachment*: without the constraint of semantic similarity, the agent may retrieve high- $Q$  memories (successful in other contexts) that are irrelevant to the current task. This confirms that semantic similarity must act as an anchor for retrieval, while RL provides the necessary gradient for optimizing memory selection.

### 5.2.3 SENSITIVITY TO RETRIEVAL SIZE ( $k_1$ AND $k_2$ ).

To investigate the impact of memory capacity on reasoning performance, we conducted an ablation study on a subset of the HLE benchmark (Computer Science/AI category). We compared two retrieval configurations: a larger recall setting ( $k_1 = 10, k_2 = 5$ ) and a compact recall setting ( $k_1 = 5, k_2 = 3$ ). As shown in Figure 7, the compact setting ( $k_1 = 5, k_2 = 3$ ) achieves superior stability compared to the larger setting, suggesting that for complex reasoning tasks like HLE, simply increasing context volume can introduce irrelevant noise that interferes with the model’s judgment. Therefore, a smaller, higher-quality set of retrieved memories is sufficient for maintaining reasoning precision.

## 5.3 DISCUSSION

In this section, we delve deeper into the mechanisms driving MEMRL’s performance, connecting empirical results back to the *Stability-Plasticity* framework proposed in Section 4.

### 5.3.1 MEMRL AS A TRAJECTORY VERIFIER.

Table 3 reveals a correlation between task structural complexity and performance gain. The gains are most profound in multi-step sequential tasks (e.g., ALFWorld +24.1 Percentage Points(pp)) compared to single-turn tasks (e.g., BigCodeBench +2.5 pp). In sequential tasks, a retrieved memory must be valid for the *entire* trajectory. Standard semantic retrieval often fetches memories that match the initial instruction but fail in later steps. By propagating the final reward backward to the memory utility  $Q$ , MEMRL effectively learns to verify the *whole trajectory*, filtering out brittle policies that look correct only on the surface.

Table 3: **Impact of Task Structure.** Comparison of Cumulative Success Rate (CSR) gains. Multi-step tasks benefit significantly more from MEMRL.

Benchmark	Interaction	MemP (%)	MEMRL (%)	Gain (pp)
ALFWorld	Multi-step	45.6	<b>69.7</b>	<b>+24.1</b>
OS Task	Multi-step	74.3	<b>81.7</b>	<b>+7.4</b>
HLE	Single-step	58.2	61.3	+3.1
BigCodeBench	Single-step	60.2	62.7	+2.5

This analysis indicates that MEMRL transcends the role of a simple retrieval enhancer to function as a *Trajectory Verifier*. Its value is maximized in tasks with complex temporal dependencies, where it learns to select memories that ensure the structural integrity of the entire interaction process.

### 5.3.2 PREDICTIVE POWER OF THE Q CRITIC

Does the learned Q-value truly reflect solution quality? Figure 8a shows a strong positive correlation (Pearson  $r = 0.861$ ) between the Critic’s estimated Q-values and empirical task success rates. The success rate increases from 21.5% in the lowest-confidence bin to 88.1% in the highest, indicating that a major driver of performance is the Critic’s ability to rank memories by their likelihood of leading to successful task completion.

Further analysis of memory composition (Figure 8b) suggests a secondary source of gain: robustness. Even in high-Q bins (0.9–1.0), the agent retains a small fraction of memories labeled as “failure” ( $\sim 12\%$ ). Rather than contradicting the Critic, this pattern is consistent with the interpretation that Q-values can capture *utility beyond binary outcomes*: some unsuccessful trajectories remain strategically useful because they encode near-correct reasoning and transferable procedural lessons.

**Case study: a high-Q “failure” memory as a transferable near-miss.** We find high-Q failure memories that summarize a minor, localized mistake and a corrective heuristic that generalizes across tasks. Here we use *near-miss* to denote trajectories that are largely correct but fail due to minor, localized errors (e.g., verification slips or tool-usage details). For example, one failure memory with  $Q = 0.9878$  corresponds to a trajectory that followed a correct approach but incorrectly treated an *empty command output* as evidence of failure. The stored reflection explicitly identifies the root cause (misinterpreting empty output), warns against the pattern (equating “no output” with failure), and recommends the correct approach (validate success via exit status, error logs, or other objective signals). When retrieved in later episodes, this single “failure” memory supports perfect downstream outcomes in our logs (15/15 successes), demonstrating that the memory is valuable precisely because it captures a fixable near-miss rather than an uninformative failure.

Taken together, these results suggest that the Critic is not merely separating “success” from “failure,” but assigning higher value to memories that provide reusable guidance—including a subset of high-utility failures that are near-miss in nature. By retaining and reusing such corrective heuristics, the agent can be more robust than simple success-replay mechanisms.

### 5.3.3 STABILITY OF MEMRL

We analyze the underlying mechanisms of MEMRL through the view of the *stability-plasticity dilemma*, examining how our framework balances the acquisition of new capabilities with the retention of established knowledge.

The superior CSR (Table 1) demonstrates that MEMRL effectively expands the agent’s solution space. Unlike heuristic baselines constrained by static similarity, which often retrieve redundant

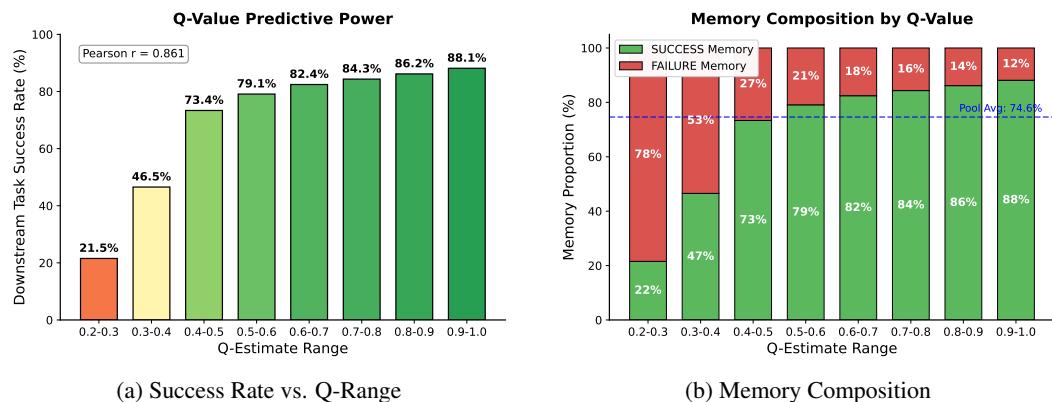


Figure 8: **Q-Value Analysis.** (a) Pearson  $r = 0.861$  confirms Critic’s predictive power. (b) Failure memories ( $\sim 12\%$ ) in high Q-bins indicate latent strategic utility.

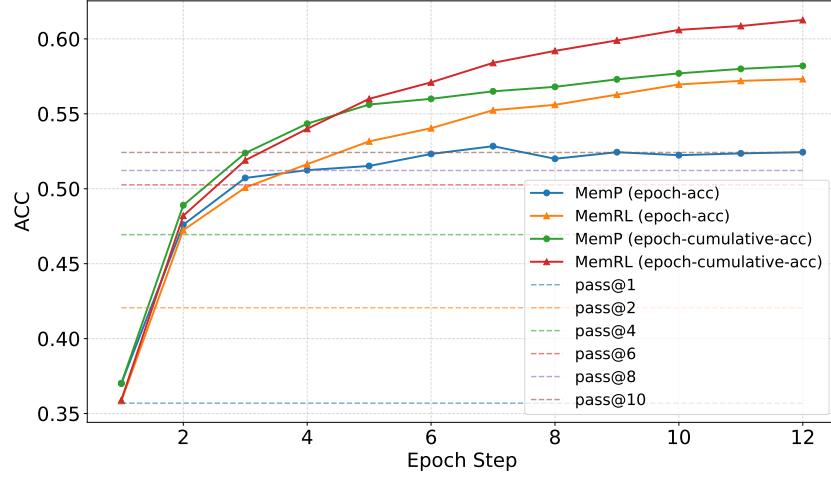


Figure 9: Epoch Acc and Cumulative ACC of MEMRL and MemP in HLE.

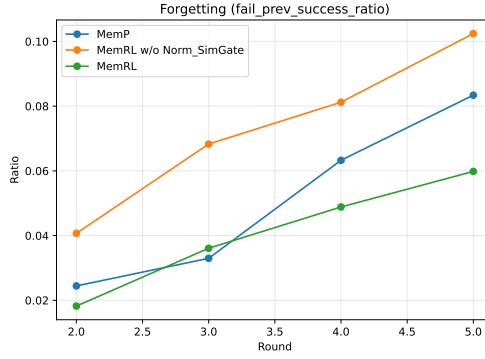


Figure 10: Forgetting Rate in HLE

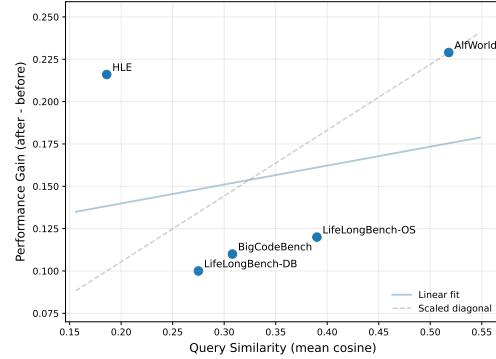


Figure 11: Similarity-based Generalization

nearest neighbors leading to repeated failures, MEMRL allows the agent to identify and reinforce non-obvious but effective strategies. This capability enables the agent to break through local optima where traditional retrieval methods stagnate.

Besides, long-term training dynamics (Figure 9) reveal a critical stability advantage. Heuristic methods like MemP suffer from a widening gap between CSR and current epoch accuracy, indicating that new explorations inadvertently overwrite effective historical policies (catastrophic forgetting). In contrast, MEMRL maintains synchronized growth. We attribute this to our theoretical guarantee. From a general MDP perspective (Eq. 3), the value update benefits from the standard Bellman contraction,  $\|\mathcal{T}Q - Q^*\|_\infty \leq \gamma\|Q - Q^*\|_\infty$  (Sutton & Barto, 2018), shrinking the error by  $\gamma$  at each step. More specifically, under our Monte Carlo style modeling (Eq. 8), this process is guaranteed by Section 4.5. Unlike heuristic ranking which may drift randomly, our approach mathematically constrains the policy to climb the variational lower bound of expected reward, ensuring the stable, non-decreasing performance observed in our experiments.

We further validate these insights using the *Forgetting Rate*, defined as the ratio of tasks that regress from success in the previous round to failure in the current round ( $\text{success} \rightarrow \text{fail}$ ). As shown in Figure 10, MEMRL achieves the lowest mean forgetting rate (0.041), outperforming the baseline MemP (0.051) and empirically confirming our analysis.

**The Necessity of Normalization and Similarity Gate.** Incidentally, the Figure 10 also highlights the necessity of our stabilization design: removing normalization and lowering similarity (w/o Norm\_SimGate) threshold causes the mean forgetting rate to spike to 0.073 due to unconstrained utility variance. This demonstrates that z-score normalization and strict similarity gating are essential for filtering noise, ensuring that the self-evolution remains stable while maximizing positive transfer.

### 5.3.4 IMPACT OF TASK SIMILARITY ON MEMORY EFFICACY.

To understand the underlying conditions where MemRL thrives, we analyze the correlation between the intra-dataset semantic similarity ( $\text{Sim}_{intra}$ ) and the absolute performance gain provided by our method ( $\Delta = \text{Acc}_{\text{MemRL}} - \text{Acc}_{\text{NoMem}}$ ).

As illustrated in Figure 11, we analyze the correlation between intra-dataset semantic similarity and the absolute performance gain ( $\Delta$ ) provided by MemRL. The linear regression trend reveals a general positive correlation: environments with higher structural repetition allow the agent to retrieve and reuse optimal policies more effectively. At the upper extreme, **ALFWorld** (similarity 0.518) acts as a strong anchor point for this trend, exhibiting the highest repetition and a corresponding maximum performance boost ( $\Delta = +0.229$ ). This confirms that for highly repetitive procedural tasks, memory serves as an effective shortcut to optimal trajectories. Following the regression line, benchmarks with moderate similarity—such as **Lifelong-OS** (0.390) and **BigCodeBench** (0.308)—cluster in the middle region, showing steady improvements ( $\Delta \approx +0.11 \sim +0.12$ ) where the agent successfully generalizes coding patterns or OS commands across related instructions.

### The HLE Anomaly: Generalization vs. Memorization.

HLE presents a unique outlier. Despite having the lowest similarity (0.186) due to its diverse, multi-disciplinary nature, it exhibits a surprisingly high runtime gain ( $0.357 \rightarrow 0.573, \Delta = +0.216$ ). This gain operates on a different mechanism than ALFWorld. In high-similarity benchmarks, MemRL succeeds via *Positive Transfer*—generalizing shared patterns to new instances. In contrast, the gain in HLE stems from *Runtime Memorization*. Since HLE questions are distinct and domain-specific, the agent relies on the Runtime Learning phase to “memorize” specific solutions to difficult problems through repeated exposure. This distinction highlights MemRL’s versatility: it supports both *pattern generalization* in structured domains and *specific knowledge acquisition* in diverse domains.

## 6 CONCLUSION

In this paper, we introduced MemRL, a novel framework that enables LLMs to self-evolve through non-parametric reinforcement learning on episodic memory. Addressing the limitations of semantic retrieval and the instability of parameter fine-tuning, MemRL treats memory retrieval as a value-based decision process. By structuring memory into Intent-Experience-Utility triplets and applying Bellman updates, the agent learns to differentiate high-value strategies from semantic noise without modifying the backbone model weights.

Our extensive evaluations across diverse domains—ranging from code generation to embodied navigation—demonstrate that MemRL significantly outperforms existing memory-augmented baselines in both runtime learning and generalization to unseen tasks. Theoretical and empirical analyses further reveal that MemRL effectively resolves the stability-plasticity dilemma: the frozen LLM provides robust reasoning, while the evolving memory utility acts as a plastic channel for adaptation. Moreover, we identify that the utility-driven retrieval mechanism functions as a trajectory verifier, enabling the agent to filter out brittle policies in complex, multi-step tasks. We hope this work establishes a new paradigm for building self-evolving agents that can continuously learn from interaction in a stable and efficient manner.

## REFERENCES

- Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection, 2023. URL <https://arxiv.org/abs/2310.11511>.
- Richard Bellman. Dynamic programming. *science*, 153(3731):34–37, 1966.
- Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory, 2025. URL <https://arxiv.org/abs/2504.19413>.
- Paul Cisek and John F Kalaska. Neural correlates of mental rehearsal in dorsal premotor cortex. *Nature*, 431(7011):993–996, 2004.

- Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellootto, and Fabrizio Silvestri. The power of noise: Redefining retrieval for rag systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR 2024, pp. 719–729. ACM, July 2024. doi: 10.1145/3626772.3657834. URL <http://dx.doi.org/10.1145/3626772.3657834>.
- A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. ISSN 00359246. URL <http://www.jstor.org/stable/2984875>.
- Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*, 2024.
- Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory. *arXiv preprint arXiv:2508.06433*, 2025. URL <https://arxiv.org/abs/2508.06433>.
- Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.
- Chunjing Gan, Dan Yang, Binbin Hu, Hanxiao Zhang, Siyuan Li, Ziqi Liu, Yue Shen, Lin Ju, Zhiqiang Zhang, Jinjie Gu, Lei Liang, and Jun Zhou. Similarity is not all you need: Endowing retrieval augmented generation with multi layered thoughts. *arXiv preprint arXiv:2405.19893*, 2024. doi: 10.48550/arXiv.2405.19893. URL <https://arxiv.org/abs/2405.19893>.
- Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1), 2023.
- Mary L Gick and Keith J Holyoak. Analogical problem solving. *Cognitive psychology*, 12(3): 306–355, 1980.
- Mary L Gick and Keith J Holyoak. Schema induction and analogical transfer. *Cognitive psychology*, 15(1):1–38, 1983.
- Stephen Grossberg. Adaptive resonance theory: How a brain learns to consciously attend, learn, and recognize a changing world. *Neural networks*, 37:1–47, 2013.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- Demis Hassabis and Eleanor A Maguire. Deconstructing episodic memory with construction. *Trends in cognitive sciences*, 11(7):299–306, 2007.
- Josue Haubrich and Karim Nader. Memory reconsolidation. *Behavioral neuroscience of learning and memory*, pp. 151–176, 2016.
- Zhengjun Huang, Zhoujin Tian, Qintian Guo, Fangyuan Zhang, Yingli Zhou, Di Jiang, and Xiaofang Zhou. Licomemory: Lightweight and cognitive agentic memory for efficient long-term reasoning. *arXiv preprint arXiv:2511.01448*, 2025.
- Khurram Javed, Haseeb Shah, Rich Sutton, and Martha White. Online real-time recurrent learning using sparse connections and selective learning. *Journal of Machine Learning Research*, 2023.
- Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pp. 6769–6781, 2020.
- James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

- Dharshan Kumaran, Demis Hassabis, and James L McClelland. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in cognitive sciences*, 20(7):512–534, 2016.
- Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review, 2018. URL <https://arxiv.org/abs/1805.00909>.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.
- Hongyu Li, Liang Ding, Meng Fang, and Dacheng Tao. Revisiting catastrophic forgetting in large language model tuning. *arXiv preprint arXiv:2406.04836*, 2024.
- Leqian Li, Dianxi Shi, Jialu Zhou, Xinyu Wei, Mingyue Yang, Songchang Jin, and Shaowu Yang. Retrieval feedback memory enhancement large model retrieval generation method. *arXiv preprint arXiv:2508.17862*, 2025a.
- Zhiyu Li, Shichao Song, Hanyu Wang, Simin Niu, Ding Chen, Jiawei Yang, Chenyang Xi, Huayi Lai, Jihao Zhao, Yezhaohui Wang, et al. Memos: An operating system for memory-augmented generation (mag) in large language models. *arXiv preprint arXiv:2505.22101*, 2025b.
- Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.
- David Lopez-Paz and Marc’Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.
- James L McClelland, Bruce L McNaughton, and Randall C O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419, 1995.
- Nicholas Metropolis and Stanislaw Ulam. The monte carlo method. *Journal of the American statistical association*, 44(247):335–341, 1949.
- Karim Nader, Glenn E Schafe, and Joseph E Le Doux. Fear memories require protein synthesis in the amygdala for reconsolidation after retrieval. *Nature*, 406(6797):722–726, 2000.
- Radford M. Neal and Geoffrey E. Hinton. *A View of the Em Algorithm that Justifies Incremental, Sparse, and other Variants*, pp. 355–368. Springer Netherlands, Dordrecht, 1998. ISBN 978-94-011-5014-9. doi: 10.1007/978-94-011-5014-9\_12. URL [https://doi.org/10.1007/978-94-011-5014-9\\_12](https://doi.org/10.1007/978-94-011-5014-9_12).
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.
- Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems, 2024. URL <https://arxiv.org/abs/2310.08560>.
- German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019.
- Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang, Mohamed Shaaban, John Ling, Sean Shi, et al. Humanity’s last exam, 2025. URL <https://arxiv.org/abs/2501.14249>.
- Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.
- Rana Salama, Jason Cai, Michelle Yuan, Anna Currey, Monica Sunkara, Yi Zhang, and Yassine Benajiba. Meminsight: Autonomous memory augmentation for llm agents. *arXiv preprint arXiv:2503.21760*, 2025.
- Daniel L Schacter and Donna Rose Addis. On the constructive episodic simulation of past and future events. *Behavioral and Brain Sciences*, 30(3):331–332, 2007.
- Daniel L Schacter, Donna Rose Addis, Demis Hassabis, Victoria C Martin, R Nathan Spreng, and Karl K Szpunar. The future of memory: remembering, imagining, and the brain. *Neuron*, 76(4):677–694, 2012.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551, 2023.
- Noah Shinn, Federico Cassano, Beck Labash, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL <https://arxiv.org/abs/2303.11366>, 1, 2023.
- Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning, 2021. URL <https://arxiv.org/abs/2010.03768>.
- David Silver and Richard S. Sutton. Welcome to the era of experience, 2025. Preprint chapter to appear in the MIT Press book *Designing an Intelligence*.
- Joykirat Singh, Raghav Magazine, Yash Pandya, and Akshay Nambi. Agentic reasoning and tool integration for llms via reinforcement learning. *arXiv preprint arXiv:2505.01441*, 2025. URL <https://arxiv.org/abs/2505.01441>.
- Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021, 2020.
- Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.
- Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2 edition, 2018. URL <http://incompleteideas.net/book/the-book-2nd.html>.
- 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. *arXiv preprint arXiv:2305.16291*, 2023.
- Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- 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.
- Tongtong Wu, Linhao Luo, Yuan-Fang Li, Shirui Pan, Thuy-Trang Vu, and Gholamreza Haffari. Continual learning for large language models: A survey, 2024. URL <https://arxiv.org/abs/2402.01364>.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.

Ye Ye. Task memory engine: Spatial memory for robust multi-step llm agents. *arXiv preprint arXiv:2505.19436*, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Zeyu Zhang, Quanyu Dai, Rui Li, Xiaohe Bo, Xu Chen, and Zhenhua Dong. Learn to memorize: Optimizing llm-based agents with adaptive memory framework. *arXiv preprint arXiv:2508.16629*, 2025.

Junhao Zheng, Xidi Cai, Qiuke Li, Duzhen Zhang, ZhongZhi Li, Yingying Zhang, Le Song, and Qianli Ma. Lifelongagentbench: Evaluating llm agents as lifelong learners, 2025. URL <https://arxiv.org/abs/2505.11942>.

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. 2025. URL <https://arxiv.org/abs/2508.16153>.

Ruiwen Zhou, Yingxuan Yang, Muning Wen, Ying Wen, Wenhao Wang, Chunling Xi, Guoqiang Xu, Yong Yu, and Weinan Zhang. TRAD: Enhancing llm agents with step-wise thought retrieval and aligned decision. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 3–13. ACM, 2024. URL <https://arxiv.org/abs/2403.06221>.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, Simon Brunner, Chen Gong, Thong Hoang, Armel Randy Zebaze, Xiaoheng Hong, Wen-Ding Li, Jean Kaddour, Ming Xu, Zhihan Zhang, Prateek Yadav, Naman Jain, Alex Gu, Zhoujun Cheng, Jiawei Liu, Qian Liu, Zijian Wang, Binyuan Hui, Niklas Muennighoff, David Lo, Daniel Fried, Xiaoning Du, Harm de Vries, and Leandro Von Werra. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions, 2025. URL <https://arxiv.org/abs/2406.15877>.

## A THEORETICAL ANALYSIS AND PROOFS

In this section, we provide the detailed derivation for the convergence of the Q-value estimation under the Exponential Moving Average (EMA) update rule, and extend the analysis to the global stability of memory utility under task distributions.

### A.1 PROOF OF THEOREM 1: CONVERGENCE OF EMA ESTIMATION

We aim to prove that for a fixed task-memory pair  $(s, m)$  with a stationary reward distribution, the Q-value estimate  $Q_t(s, m)$  converges in expectation to the true mean reward  $\beta(s, m)$ .

#### Assumptions.

1. **Stationary Reward.** The reward  $r_t$  at step  $t$  is drawn from a distribution with a constant mean  $\beta(s, m) = \mathbb{E}[r_t | s, m]$  and finite variance  $\sigma^2$ .
2. **Update Rule.** The utility is updated via the linear EMA rule with learning rate  $\alpha \in (0, 1)$ :

$$Q_{t+1} = (1 - \alpha)Q_t + \alpha r_t.$$

**Derivation of Error Dynamics.** Let  $e_t \triangleq Q_t - \beta(s, m)$  be the estimation error at time step  $t$ . Substituting  $Q_t = e_t + \beta(s, m)$  into the update rule:

$$\begin{aligned} e_{t+1} + \beta(s, m) &= (1 - \alpha)(e_t + \beta(s, m)) + \alpha r_t \\ e_{t+1} &= (1 - \alpha)e_t + (1 - \alpha)\beta(s, m) + \alpha r_t - \beta(s, m) \\ e_{t+1} &= (1 - \alpha)e_t + \beta(s, m) - \alpha\beta(s, m) - \beta(s, m) + \alpha r_t \\ e_{t+1} &= (1 - \alpha)e_t + \alpha(r_t - \beta(s, m)). \end{aligned} \tag{14}$$

**Convergence Analysis.** We define  $\mathcal{F}_t$  as the filtration (history) up to time  $t$ . Taking the conditional expectation of Eq. 14 given  $\mathcal{F}_t$ :

$$\mathbb{E}[e_{t+1} | \mathcal{F}_t] = (1 - \alpha)e_t + \alpha(\underbrace{\mathbb{E}[r_t | \mathcal{F}_t]}_{\beta(s, m)} - \beta(s, m)) = (1 - \alpha)e_t.$$

By the Law of Iterated Expectations, taking the full expectation yields:

$$\mathbb{E}[e_{t+1}] = \mathbb{E}[\mathbb{E}[e_{t+1} | \mathcal{F}_t]] = (1 - \alpha)\mathbb{E}[e_t].$$

Iterating this recurrence relation from  $t = 0$ :

$$\mathbb{E}[e_t] = (1 - \alpha)^t \mathbb{E}[e_0].$$

Since  $0 < \alpha < 1$ , we have  $|1 - \alpha| < 1$ . Consequently:

$$\lim_{t \rightarrow \infty} \mathbb{E}[e_t] = \mathbb{E}[e_0] \cdot \lim_{t \rightarrow \infty} (1 - \alpha)^t = 0. \tag{15}$$

This proves that the estimator is unbiased in the limit, i.e.,  $\lim_{t \rightarrow \infty} \mathbb{E}[Q_t] = \beta(s, m)$ .  $\square$

### A.2 BOUNDED VARIANCE AND GLOBAL STABILITY

In this section, we provide the formal derivation for the variance bound of the estimator  $Q_t$ . We explicitly derive the finite-time variance formula via recursive unrolling and prove its asymptotic convergence, demonstrating how Phase-A clustering contributes to global stability.

**Derivation of the Variance Bound.** Let  $\sigma^2 \triangleq \text{Var}(r_t|s, m)$  be the variance of the reward signal, assumed to be finite. The EMA update rule is given by:

$$Q_{t+1} = (1 - \alpha)Q_t + \alpha r_t.$$

Since the reward  $r_t$  (current noise) is statistically independent of the current estimate  $Q_t$  (which is determined by history  $\mathcal{F}_{t-1}$ ), the variance of the sum is the sum of the variances:

$$\begin{aligned} \text{Var}(Q_{t+1}) &= \text{Var}((1 - \alpha)Q_t + \alpha r_t) \\ &= (1 - \alpha)^2 \text{Var}(Q_t) + \alpha^2 \sigma^2. \end{aligned}$$

Let  $v_t \triangleq \text{Var}(Q_t)$ . We obtain a linear recurrence relation  $v_{t+1} = (1 - \alpha)^2 v_t + \alpha^2 \sigma^2$ .

**Recursive Unrolling.** To solve for  $v_t$ , we expand the recurrence relation backward from step  $t$ :

$$\begin{aligned} v_t &= (1 - \alpha)^2 v_{t-1} + \alpha^2 \sigma^2 \\ &= (1 - \alpha)^2 [(1 - \alpha)^2 v_{t-2} + \alpha^2 \sigma^2] + \alpha^2 \sigma^2 \\ &= (1 - \alpha)^4 v_{t-2} + \alpha^2 \sigma^2 [1 + (1 - \alpha)^2] \\ &\quad \vdots \\ &= (1 - \alpha)^{2t} v_0 + \alpha^2 \sigma^2 \sum_{k=0}^{t-1} ((1 - \alpha)^2)^k. \end{aligned} \tag{16}$$

Eq. 16 explicitly shows that the variance at time  $t$  consists of two components: the decayed initial variance (first term) and the accumulated noise variance (second term).

**Asymptotic Convergence.** As  $t \rightarrow \infty$ , since the learning rate  $\alpha \in (0, 1)$ , the term  $(1 - \alpha)^{2t}$  vanishes. The summation term is a geometric series  $\sum_{k=0}^{\infty} r^k = \frac{1}{1-r}$  with ratio  $r = (1 - \alpha)^2$ . Thus:

$$\lim_{t \rightarrow \infty} v_t = \alpha^2 \sigma^2 \cdot \frac{1}{1 - (1 - \alpha)^2}.$$

Evaluating the denominator:

$$1 - (1 - \alpha)^2 = 1 - (1 - 2\alpha + \alpha^2) = 2\alpha - \alpha^2 = \alpha(2 - \alpha).$$

Substituting this back yields the tight variance bound:

$$\limsup_{t \rightarrow \infty} \text{Var}(Q_t) = \frac{\alpha^2 \sigma^2}{\alpha(2 - \alpha)} = \frac{\alpha}{2 - \alpha} \sigma^2. \tag{17}$$

**Connection to Phase-A Clustering.** This result provides the theoretical justification for the stability of MEMRL. While tasks within a memory cluster  $\mathcal{S}(m) \triangleq \{s | \text{sim}(s, z_m) > \tau_A\}$  may vary, the Smoothness Assumption implies their rewards are drawn from a distribution with bounded variance  $\sigma_{\mathcal{S}(m)}^2$ . The derived bound  $\frac{\alpha}{2 - \alpha} \sigma_{\mathcal{S}(m)}^2$  guarantees that the memory utility  $Q(m)$  will not diverge but will instead oscillate within a controlled range around the true expected utility. This mechanism effectively filters out high-frequency noise from diverse task instances while retaining the stable generalized value.

## B THEORETICAL ANALYSIS: CONVERGENCE VIA VARIATIONAL INFERENCE

In this section, we provide a theoretical foundation for MEMRL, demonstrating that our retrieval strategy and update rules guarantee the convergence of value estimation.

### B.1 THE CONVERGENCE OBJECTIVE

Our ultimate goal is to ensure that the estimated utility  $Q(m)$  converges to the true expected return of memory  $m$ . This target value is defined as:

$$\lim_{t \rightarrow \infty} \mathbb{E}[Q_t(m)] = \mathbb{E}[r|m] = \sum_{s \in \mathcal{S}(m)} \underbrace{\mathbb{E}[r|s, m]}_{\text{Stationary}} \underbrace{\Pr(s|m)}_{\text{Retrieve-Dependent}}. \tag{18}$$

The challenge lies in the term  $\Pr(s|m)$ —the probability that a specific state  $s$  triggers the retrieval of  $m$ . This distribution depends on the retrieval policy  $\mu_t(m|s)$ , which itself evolves during training, creating a circular dependency that threatens stability.

## B.2 VARIATIONAL OBJECTIVE WITH TRUST REGION

To resolve this, we formulate the problem as maximizing a **global variational objective**  $\mathcal{J}(\mu, Q)$ . This objective serves as a tractable lower bound for the global expected return defined in Eq. 18, balanced by a semantic trust region:

$$\mathcal{J}(\mu, Q) = \mathbb{E}_{s \sim \mathcal{D}} \left[ \underbrace{\sum_{m \in \mathcal{S}(s)} \mu(m|s) Q(s, m)}_{\text{Expected Utility } \approx \mathbb{E}[Q_t(m)]} - \frac{1}{\beta} \underbrace{D_{\text{KL}}(\mu(\cdot|s) \parallel \pi_{\text{sim}}(\cdot|s))}_{\text{Semantic Trust Region}} \right] \quad (19)$$

Here, the first term directly corresponds to the expected utility  $\mathbb{E}[Q_t(m)]$  we aim to converge, while  $\pi_{\text{sim}}$  represents the fixed semantic prior (derived from Phase-A). The KL-divergence term acts as a regularizer crucial for two reasons:

1. **Trust Region:** It constrains the policy to the support set  $\mathcal{S}$ , preventing the agent from retrieving high-Q but semantically irrelevant memories (out-of-distribution errors).
2. **Regularization:** It stabilizes the learning dynamics during the “cold start” phase when Q-estimates are noisy.

## B.3 OPTIMIZATION VIA GENERALIZED EXPECTATION-MAXIMIZATION (GEM)

We treat the optimization of  $\mathcal{J}$  as a GEM process, alternating between policy improvement and value evaluation:

**E-Step (Policy Optimization).** Fixing  $Q_t$ , we find the optimal policy  $\mu^*$  that maximizes  $\mathcal{J}$ . The closed-form solution is the Boltzmann distribution (Levine, 2018):

$$\mu^*(m|s) \propto \pi_{\text{sim}}(m|s) \exp(\beta Q_t(s, m))$$

By taking the logarithm, we recover the specific scoring function used in our **Phase-B Retrieval** (Eq. 6):

$$\log \mu^*(m|s) \propto \underbrace{\log \pi_{\text{sim}}(m|s)}_{\approx \text{sim}(s, m)} + \beta Q_t(s, m)$$

This proves that our heuristic combination of similarity and Q-value is mathematically equivalent to the optimal policy under the variational objective.

**M-Step (Policy Evaluation via Error Minimization).** While the E-step improves the policy based on current estimates, the M-step ensures these estimates are grounded in reality. Fixing the policy  $\mu_{t+1}$ , our goal is to align the variational parameter  $Q$  with the true environmental returns. We formulate this as minimizing the Mean Squared Error (MSE) between the estimated utility and the observed reward target  $y = r$  (in our Monte Carlo style modeling):

$$\min_Q \mathcal{L}(Q) = \mathbb{E}_{\tau \sim \mu_{t+1}} \left[ \frac{1}{2} (y - Q(s, m))^2 \right]$$

Minimizing this error is critical because it tightens the variational bound: it ensures that the expectation term  $\mathbb{E}[Q]$  in the global objective  $\mathcal{J}$  (Eq. 19) converges to the true expected return  $\mathbb{E}[r]$ . The update rule used in our framework (Eq. 8) corresponds exactly to a Stochastic Gradient Descent (SGD) step on this objective:

$$Q_{t+1}(s, m) \leftarrow Q_t(s, m) - \alpha \nabla_Q \mathcal{L}(Q) = Q_t(s, m) + \alpha(y - Q_t(s, m))$$

By iteratively minimizing  $\mathcal{L}(Q)$ , the M-step propagates the environmental feedback into the utility estimates, ensuring that the subsequent E-step optimization occurs on a reliable value landscape.

#### B.4 PROOF OF CONVERGENCE

By the *Monotonic Improvement Theorem* of GEM (Neal & Hinton, 1998), the sequence  $(\mu_t, Q_t)$  is guaranteed to converge to a stationary point  $(\mu^*, Q^*)$ . At stationarity, the policy stabilizes  $(\mu_{t+1} \approx \mu_t)$ , which implies that the inverse retrieval probability  $\Pr(s|m)$  becomes **time-invariant**:

$$\Pr(s|m) = \frac{\mu^*(m|s) \Pr(s)}{\sum_{s'} \mu^*(m|s') \Pr(s')}$$

Consequently, the “Retrieve-Dependent” term in Eq.18 is anchored. With a fixed data distribution, the standard Bellman contraction property ensures that  $Q_t(m)$  converges to the unique fixed point:

$$\lim_{t \rightarrow \infty} Q_t(m) \rightarrow \mathbb{E}_{\mu^*}[r|m] \quad (20)$$

Thus, our framework theoretically guarantees that the memory values converge to the true expected returns under the optimal retrieval policy.