YET ANOTHER AGENT MEMORY SYSTEM? HIPPO: TRAINING-FREE AGENT SELF-EVOLUTION VIA HIPPOCAMPAL REPLAY

Anonymous authors
Paper under double-blind review

ABSTRACT

Motivated by recursive self-improvement (RSI), we study how LLM agents can learn through external memory without updating model parameters. Extracting useful experience from a single attempt is difficult: flawed reasoning can produce success, while poor execution can obscure a sound strategy. Preserving its value over time is also difficult: our accumulation baseline without consolidation initially improves but eventually performs below a no-memory baseline. Our key idea is to compare attempts at the same task to identify reusable decisions and refine when they apply. We introduce Hippo, a memory framework inspired by selective hippocampal encoding and replay that learns from these comparisons when shortcomings are diagnosed. It compares the resulting experience with existing memory to merge overlaps and resolve conflicts through explicit applicability conditions. This allows prior guidance to be revised without storing every lesson as a separate entry. Experiments on SWE-bench Verified, WebArena, and Mind2Web show performance gains in all three settings and substantially fewer retained experiences than the accumulation baseline. The gains extend across the evaluated base models of varying capabilities, indicating that memory-based adaptation remains useful even for highly capable models.

1 INTRODUCTION

Self-evolving agents seek to turn experience from past tasks into better decisions on future tasks (Gao et al., 2026). This objective has motivated research on recursive self-improvement, including agents that revise their own code in response to execution feedback (Zhang et al., 2026a). External memory offers a form of *gradient-free learning* without changes to model parameters (Shinn et al., 2023; Zhao et al., 2024). Past experience becomes part of the context for later decisions, so the agent's adaptation depends on what it retains and how it revises that experience.

Prior work has established several ways to represent, update, and reuse experience. ReasoningBank's MaTTS also compares attempts to form memory, while its reported consolidation implementation appends new items without further pruning or merging (Ouyang et al., 2026).Vanilla-IB serves as a baseline for examining how interaction steps relate to token usage and task performance (Hajimiri et al., 2026).We focus on an update policy in which identified shortcomings determine whether to form new experience, what it should contain, and how it should revise prior guidance.

Continued adaptation, however, does not per se guarantee continued improvement. An exmaple is illustrated in Figure 1. In our experiments, we examine an accumulation baseline that adds experience from both successful and failed attempts without consolidation. With this baseline, performance follows an inverted-U pattern as memory grows: it improves at first, then declines and eventually falls below the no-memory baseline.

This observation motivates our central question: *how should an agent decide what experience to retain and how that experience should revise existing memory?*

An execution trace does not directly reveal what the agent should learn from it. A single failed attempt may reflect a flawed strategy or poor execution of a valid plan. A correct outcome can also conceal incorrect reasoning. A reflection may therefore extract a flawed lesson that later re-

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.

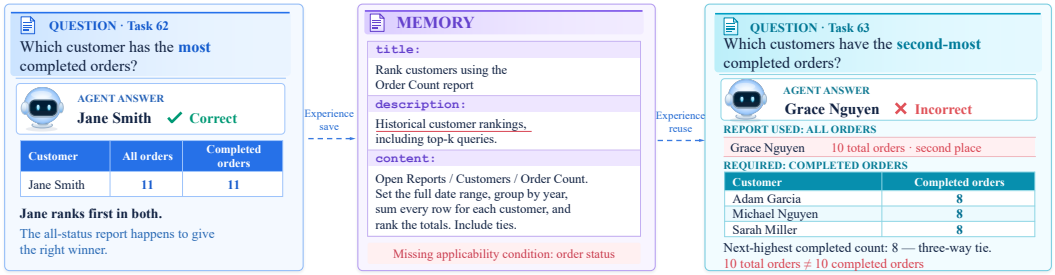

Figure 1: Correct outcomes can mask flawed reasoning, leading to erroneous or overly general lessons. Once retained in memory, these lessons may be repeatedly retrieved and misguide decisions across subsequent tasks.

trieval repeatedly presents as a trusted precedent. Even a sound lesson needs to capture the decision behind the observed actions and the conditions under which that decision applies. Task-specific action sequences may transfer poorly, while generic advice may offer little practical guidance. New information may also duplicate or contradict what the agent already retains (Chhikara et al., 2025).

We propose Hippo, a memory framework for self-evolving agents. The design draws on selective hippocampal encoding of salient episodes and replay of past experience during consolidation (Lisman & Grace, 2005; Kumaran et al., 2016). These processes motivate a separation between episodic reflections on individual attempts and persistent agent memory that integrates experience across tasks. Hippo reflects only on attempts judged to fail or succeed for the wrong reasons. We refer to these cases as *surprise*.

Hippo uses two comparisons to determine how memory should change. It first compares multiple attempts at the same task, sampled under the same frozen policy and retrieved memory, to distill candidate experience from their decisions and outcomes. It then compares this experience with related memories to merge overlaps and resolve conflicts through explicit applicability conditions. These comparisons determine both what can be learned from the current task and how it should revise prior guidance. A revision can therefore change existing guidance without adding another entry. The revised memory guides decisions on subsequent tasks. Experience from those tasks, in turn, provides evidence for further memory updates.

We evaluate Hippo on software issue resolution with SWE-bench Verified (Jimenez et al., 2024; OpenAI, 2024), interactive web task completion in WebArena (Zhou et al., 2024), and next-action prediction from recorded webpage states in Mind2Web (Deng et al., 2023). Hippo improves task performance in all three settings and retains substantially fewer experiences than the accumulation baseline. Experiments show gains across base models with different levels of capability. These gains extend to highly capable models such as GPT-5.6, which indicates that memory-based adaptation remains useful even for agents with strong base capabilities. Our main contributions are:

- We quantify how repeated attempts improve the chance of task success even without memory updates. This insight motivates direct retries as a baseline for future evaluations of self-evolving agents under comparable budgets, feedback, and success criteria.
- We propose Hippo, a framework for memory-based self-evolution that couples selective experience induction from multiple attempts at the same task with maintenance of existing memory. Experiments across task settings and base models demonstrate its effectiveness.
- We identify a failure mode of memory-based self-evolution: continued experience accumulation without consolidation can erode initial gains and eventually reduce performance below the no-memory baseline.

2 Related Works

Generative Agents (Park et al., 2023) use reflection to derive higher-level memories from stored observations. Reflexion records reflections on task feedback for subsequent trials (Shinn et al., 2023), while Agent Workflow Memory abstracts recurring routines into reusable workflows (Wang

et al., 2025). However, Zhao et al. (2026) find that task success does not consistently decline when condensed experience is replaced with meaningless symbols. This result also holds in their ReasoningBank evaluation, where condensed experience is *de facto* the sole source of historical task information available to the agent.

ReasoningBank's parallel MaTTS compares multiple trajectories for the same query to distill reasoning strategies (Ouyang et al., 2026). Hippo ties experience induction to task-level prediction gap and to distill and revise semantic knowledge.MemGPT manages information across in-context and external storage (Packer et al., 2023). ExpeL revises or removes stored insights (Zhao et al., 2024). Mem0 extracts and updates salient conversational information (Chhikara et al., 2025), while A-MEM updates the descriptions and attributes of related memories (Xu et al., 2025). ReasoningBank's reported consolidation implementation directly appends new entries without further pruning or merging (Ouyang et al., 2026). MemRL uses runtime reinforcement learning to refine retrieval utilities while its LLM remains frozen (Zhang et al., 2026b). Hippo compares new experience with related memories to merge overlaps and resolve conflicts through explicit applicability conditions.Memory-R1 and MEM1 use reinforcement learning to train memory policies (Yan et al., 2025; Zhou et al., 2025). Hippo instead updates external memory while model parameters fixed.

3 HIPPO

Architecturally, Hippo updates its memory from multiple attempts at the same task through four components. (i) The multi-rollout exploration module runs the agent N times with the same frozen policy and retrieved memory. (ii) The surprise-driven episodic buffer records reflections from attempts that fail or succeed for the wrong reasons, acting as a hippocampal episodic buffer. (iii) The memory evolution engine draws on hippocampal replay to transform these reflections into the final agent memory. It compares the rollouts to distill reusable experience, then checks this experience against existing memory to merge overlaps and resolve conflicts. (iv) The retrieval and injection module selects relevant memories and provides them to the agent for later tasks. Figure 2 shows the overall architecture.

3.1 SURPRISE-DRIVEN EPISODIC REFLECTION

A core challenge in self-evolving memory systems is to decide what content deserves retention. Models of the hippocampal–VTA loop link novelty signals to enhanced long-term encoding (Lisman & Grace, 2005), while human imaging studies show hippocampal responses to violations of learned event sequences (Kumaran & Maguire, 2006). In self-evolving systems, simply accumulating routine experiences often steers the agent off-course and degrades performance over time ((Section 4.2)). We use a judge $\mathcal{J}$ to evaluate each rollout and determine whether to extract an episodic reflection:

$$L_1 = \begin{cases} \mathcal{G}(q, \tau, \delta), & \text{if } \mathcal{J}(q, \tau, r) = \text{surprise}, \\ \emptyset, & \text{otherwise.} \end{cases} \tag{1}$$

Here, q is the task, τ is the execution trace. Here, r denotes feedback when available and is empty otherwise. On SWE-bench Verified, the judge estimates success from the execution trace. δ is the diagnosis returned by $\mathcal{J}$, and $\mathcal{G}$ is the reflection extractor. The judge returns surprise unless the task succeeds and the agent's reasoning is correct. Thus, rollouts that are right for the wrong reasons still count as surprises and yield episodic reflections L_1.

3.2 MULTI-ROLLOUT EXPLORATION SCAFFOLD

A single failed trajectory may not reveal whether the approach was flawed or a valid plan was poorly executed. Models of causal induction emphasize comparisons across relevant cases (Cheng & Novick, 1990). Multiple attempts under the same conditions provide a broader evidence base and reduce the risk of overfitting a lesson to one trajectory. We therefore apply the episodic buffer of Section 3.1 to N rollouts of the same task, sampled in parallel from the frozen agent policy π with temperature sampling and conditioned on the same retrieved memory $\mathcal{M}$:

$$\mathcal{E}(q) = \left\{ (\tau_i, r_i, \delta_i, L_1^{(i)}) \right\}_{i=1}^{N}, \qquad \tau_i \sim \pi(\cdot \mid q, \mathcal{M}). \tag{2}$$

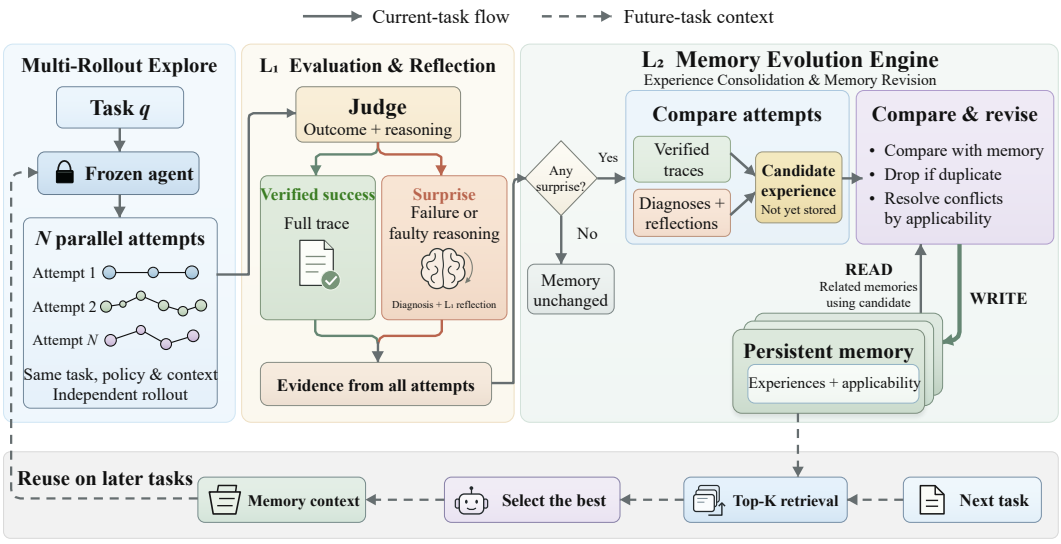

Figure 2: Overview of Hippo. Hippo combines surprise-driven episodic reflection with memory consolidation inspired by hippocampal replay, while keeping the agent policy fixed.

Each rollout is judged and reflected on independently by Eq. equation 1, and executes concurrently in its own replica of the space so that the N outcomes remain independent measurements. Together, these rollouts provide broader experience and isolate patterns that generalize across trajectories. The resulting experience set $\mathcal{E}(q)$ serves as the input to the memory evolution engine, which produces the final agent memory L_2 (Section 3.3).

3.3 Memory Evolution Engine

A central challenge for self-evolving agent memory is to remain useful over long task streams. Our experiments show that evaluated basedline performance follows an inverted-U curve as memory grows: it improves at first, then falls below the no-memory baseline (Section 4.2). We further observe three problems: (i) (i) a single reflection can extract a flawed lesson from limited evidence, and repeated retrieval may carry that error into later tasks; (ii) reflections may retain surface actions rather than the decisions behind them, and (iii) duplicate or conflicting experiences make retrieval unreliable. Thus, For a self-evolving agent that continuously tackle new tasks, it requires an active mechanism to revise, merge, and prune past experiences.

Hippocampal replay provides biological inspiration for this process. Wilson & McNaughton (1994) show that hippocampal activity patterns associated with waking experience can reactivate during subsequent sleep. In complementary learning systems theory, reactivation supports the gradual integration of new experience with existing neocortical knowledge (McClelland et al., 1995; Kumaran et al., 2016). This account inspires Hippo's two-stage update. It first contrasts the N rollouts in $\mathcal{E}(q)$ to distill a candidate experience. It then compares the candidate against existing memory to merge overlapping experiences and resolve conflicts. Together, these stages transform episodic reflections L_1 into the final agent memory L_2.

Contrastive Experience Induction. We compare attempts at the same task to derive a candidate experience m_t: a decision together with its conditions of use. The induction operator $\mathcal{D}$ performs this comparison with an LLM after selecting the evidence supplied by each rollout.

A verified success has a successful outcome and reasoning accepted by the judge. Let n_c count verified successes among the N attempts at q_t, and set $\rho_t = n_c/N$. From $\mathcal{E}(q_t)$, $\mathcal{D}$ selects full traces for verified successes and judge diagnoses with L_1 reflections for all other attempts, including successes for the wrong reasons. For $\rho_t < 1$, the process is

$$\big(q_t, \mathcal{E}(q_t), \rho_t\big) \xrightarrow{\mathcal{D}} \underbrace{m_t}_{\text{candidate experience}} . \tag{3}$$

The LLM is not given the ground-truth target directly. When $\rho_t = 1$, induction is skipped and memory remains unchanged.

For batches that trigger induction, $\mathcal{D}$ follows four cases. (i) With a verified-success majority, it turns the minority's errors into cautions. (ii) With a nonempty verified-success minority, it identifies decisions that distinguish successful attempts. (iii) With an even split, it extracts both a successful strategy and a failure trap. (iv) With no verified successes, it summarizes failure patterns documented in the diagnoses and reflections. In the last case, it is instructed to restrict guidance to actions supported by those inputs. The resulting candidate is passed to memory consolidation.

Memory Consolidation. Let $L_{2,t}$ be the persistent L_2 bank, represented as a set of identified entries, and $\mathcal{R}_t \subseteq L_{2,t}$ the entries retrieved by similarity to a nonempty candidate m_t. The language-model comparator $\mathcal{C}$ checks for redundancy and conflicting guidance, then specifies additions Δ_t^+ and deletions $\Delta_t^- \subseteq \mathcal{R}_t$:

$$(\Delta_t^+, \Delta_t^-) = \mathcal{C}(m_t, \mathcal{R}_t),$$
$$L_{2,t+1} = \underbrace{(L_{2,t} \setminus \Delta_t^-)}_{\text{unchanged entries}} \cup \underbrace{\Delta_t^+}_{\text{new entries}} . \tag{4}$$

Duplicate candidates are discarded. Overlaps and conflicts are addressed by making applicability conditions explicit in the new entries. When new guidance replaces an old entry, the engine adds the replacement and deletes the old entry. Entries outside the deletion set retain their original text. Section 4.6 compares this update policy with append-only updates and in-place rewriting.

3.4 Memory Retrieval and Injection

Semantic similarity helps identify related experiences, but does not determine whether their guidance applies to the current task. Hippo therefore uses a two-stage procedure that combines embedding retrieval with agent-based selection. For each task q, we first retrieve the top-K candidate experiences from L_2 by embedding similarity. The agent then evaluates these candidates against the task requirements and selects the applicable experiences as $\mathcal{M}$. We inject $\mathcal{M}$ into the agent's context before execution, and all N rollouts of q share the same selected memory. At each step, the agent selects an action under the frozen policy π, conditioned on the current task state and $\mathcal{M}$.

We compare this procedure with BM25 (Robertson & Zaragoza, 2009), BGE-based top-K retrieval (Xiao et al., 2024), hybrid sparse–dense retrieval via reciprocal rank fusion (Cormack et al., 2009), and direct LLM-based memory selection inspired by RankGPT (Sun et al., 2023). On recorded WebArena queries with a matching experience available, Hippo achieves a selection rate of 75.2%. Appendix B.1 provides the evaluation protocol and detailed comparisons.

4 Experiments and Results Analysis

Benchmarks and models. We evaluate Hippo on SWE-bench Verified (Jimenez et al., 2024; OpenAI, 2024) for coding task, WebArena (Zhou et al., 2024) for interactive web tasks, and Mind2Web (Deng et al., 2023) for offline next-action prediction. We use four base models with fixed parameters: GPT-5.6 Sol, Claude Haiku 4.5, Gemini 3 Flash Preview, and Qwen3.5-27B. The result tables list the models used for each benchmark.

Baselines. We compare Hippo with No Memory, Vanilla-IB (Hajimiri et al., 2026), AWM (Wang et al., 2025), and ReasoningBank (Ouyang et al., 2026). No Memory and Vanilla-IB operate without memory. We use the official implementations of AWM and ReasoningBank and enable MaTTS for ReasoningBank. On WebArena, both methods receive official outcomes and reference answers during memory construction. All methods and ablated variants use at most 30 interaction steps per rollout, except Vanilla-IB, which uses 50. Appendix D.2 describes feedback access, and Appendix A.1 reports interaction lengths.

Evaluation. We report resolution rate on all 500 SWE-bench Verified tasks, task success rate on WebArena, and step success rate on Mind2Web. For the main results, we average scores over eight

attempts per task, then across tasks. We report the mean ± sample standard deviation across three complete runs with different random seeds. Further details are provided in Appendix C.

4.1 Performance among different benchmarks

Table 1: Performance comparison on WebArena. Blue and yellow highlight the best and second-best results, respectively, in each column within each model group.

Method	Shopping (187)	Admin (182)	GitLab (180)	Reddit (106)	Map (109)	Overall (764)
GPT-5.6 Sol						
No Memory	53.4 ±1.1	78.9 ±0.6	67.9 ±0.8	56.8 ±1.2	59.3 ±1.2	64.2
Vanilla-IB	54.0 ±0.6	80.5 ±1.2	68.5 ±1.0	57.6 ±0.8	60.1 ±0.8	65.1
AWM	52.7 ±3.4	78.7 ±3.4	67.7 ±3.0	57.8 ±2.1	58.8 ±2.3	64.0
ReasoningBank	54.3 ±1.3	79.2 ±2.5	67.7 ±3.2	56.3 ±2.8	59.1 ±3.2	64.4
Hippo	64.2 ±0.6	87.2 ±1.3	73.3 ±3.0	62.5 ±0.8	73.4 ±1.0	72.9
Claude Haiku 4.5						
No Memory	39.2 ±1.2	31.4 ±1.0	33.1 ±0.8	34.8 ±0.6	28.7 ±0.6	33.8
Vanilla-IB	40.8 ±1.2	37.0 ±0.8	34.6 ±1.2	36.4 ±1.0	30.8 ±0.7	36.4
AWM	38.5 ±3.8	29.9 ±3.8	32.9 ±3.4	36.7 ±4.4	27.4 ±2.2	33.3
ReasoningBank	39.8 ±2.2	33.2 ±2.2	33.6 ±2.9	35.3 ±2.8	29.0 ±2.9	34.6
Hippo	50.1 ±1.5	39.8 ±1.0	38.6 ±1.0	40.6 ±1.2	42.3 ±0.8	42.5
Gemini 3 Flash Preview						
No Memory	46.2 ±0.4	48.3 ±1.0	26.8 ±1.2	44.3 ±0.7	26.3 ±1.0	39.0
Vanilla-IB	46.2 ±1.2	54.3 ±1.7	29.1 ±1.3	47.4 ±0.8	29.5 ±0.5	41.9
AWM	42.0 ±1.2	46.7 ±0.5	24.7 ±1.2	46.5 ±1.7	24.1 ±1.8	37.1
ReasoningBank	46.3 ±0.3	49.3 ±0.6	23.9 ±0.4	48.3 ±0.6	30.9 ±0.5	39.8
Hippo	53.8 ±0.6	56.9 ±0.8	32.5 ±0.5	50.8 ±0.5	34.8 ±0.8	46.4
Qwen3.5-27B						
No Memory	42.0 ±1.0	41.7 ±0.5	24.3 ±0.4	40.1 ±0.8	30.1 ±1.2	35.8
Vanilla-IB	43.5 ±0.6	47.4 ±1.3	25.9 ±0.7	41.8 ±0.6	31.8 ±1.1	38.4
AWM	41.1 ±3.6	40.2 ±3.7	24.0 ±3.6	42.0 ±4.2	30.0 ±2.6	35.4
ReasoningBank	43.5 ±1.0	43.3 ±2.1	24.9 ±2.1	42.4 ±2.6	31.9 ±0.9	37.3
Hippo	51.7 ±1.2	49.2 ±1.0	29.2 ±2.3	45.2 ±1.0	42.6 ±0.7	43.6

Hippo achieves the best overall performance across all evaluated backbones on WebArena in Table 1 and SWE-bench in Table 2, and leads across all three generalization splits on Mind2Web in Figure 3. The benefits of AWM and ReasoningBank vary across models and task environments, indicating that adding memory does not necessarily improve performance. The more consistent gains observed with Hippo align with its design goal of improving experience reuse through selective induction and applicability control.

Table 2: Performance comparison on SWE-bench. Blue and yellow highlight the best and second-best results, respectively, in each model column.

Method	*GPT-5.6 Sol*	*Claude Haiku 4.5*	*Gemini 3 Flash Preview*	*Qwen3.5 27B*
No Memory	81.5 ±0.4	62.6 ±0.2	74.2 ±0.4	68.8 ±0.6
Vanilla-IB	82.6 ±0.6	63.2 ±0.2	75.2 ±0.8	69.2 ±0.5
AWM	79.9 ±1.5	61.0 ±2.5	73.7 ±1.9	67.4 ±2.4
ReasoningBank	82.5 ±1.9	62.8 ±1.5	75.0 ±2.1	70.1 ±2.4
Hippo	88.5 ±1.2	70.7 ±1.4	81.6 ±1.4	76.5 ±1.8

The task-level results on WebArena reveal substantial variation in the benefits of memory augmentation. Hippo generally achieves larger gains on Shopping and Map, while improvements on GitLab and Reddit are more modest. Despite these differences, Hippo outperforms No Memory on every website across all evaluated backbones and surpasses the other memory methods in most settings.

On Mind2Web, Hippo leads across the Cross-Task, Cross-Website, and Cross-Domain splits with Gemini 3 Flash Preview. We omit further memory-augmented comparisons with GPT-5.6 Sol because its No Memory performance already exceeds 90% under our evaluation setting.

Notably, even with GPT-5.6 Sol, Hippo improves the overall WebArena score by 8.7% and outperforms No Memory on all five websites. It also improves this backbone's performance on SWE-bench, alongside consistent gains across the other evaluated models. These results indicate that strong model capabilities still leave room for improvement through experience reuse, providing em-

pirical support that Hippo can learn useful guidance from interaction histories and improve agent performance without updating model parameters.

4.2 Performance during Continued Learning

Long-term self-evolution requires experience that remains useful across many tasks and repeated memory updates, growing memory bank alone does not demonstrate continued learning.. Prior work shows that later adaptation can weaken earlier capabilities despite fixed model parameters (Yu et al., 2026). We investigate this behavior using GPT-5.6 Sol on a collection of 450 tasks from SWE-bench Verified and SWE-bench Pro (Deng et al., 2025), with all methods following the same task order. At each checkpoint, we evaluate performance against the no-memory baseline on the first 50 tasks and the same fixed set of 50 unseen tasks, assessing retention and transfer, respectively. Evaluation does not modify the memory used for subsequent learning.

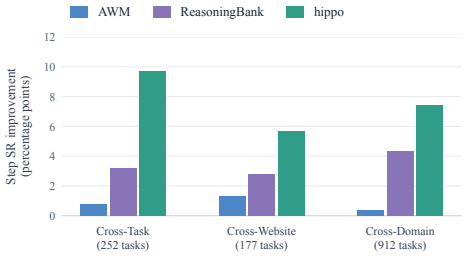

Figure 3: Improvements in step success rate over the No Memory baseline for different methods across the three Mind2Web evaluation splits.

AWM and ReasoningBank initially benefit from memory, but eventually underperform their corresponding no-memory baselines on both task sets (Figure 4(a)). This decline occurs despite continued growth in their memory banks (Figure 4(b)). In this sequence, performance crosses below the baseline earlier on unseen tasks than on the first 50 tasks.

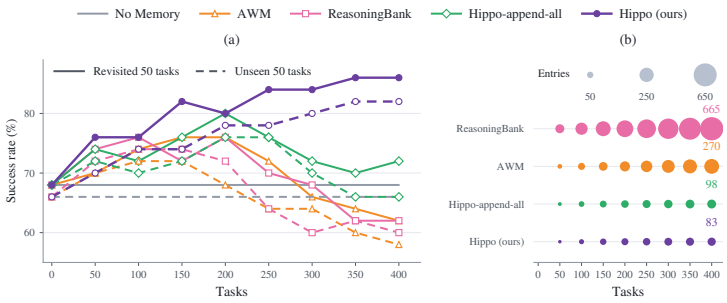

Figure 4: Long-horizon performance and memory size with GPT-5.6 Sol. (a) Success rates on the first 50 tasks and a fixed set of 50 unseen tasks, evaluated at each checkpoint. (b) Retained memory entries, indicated by circle size.

Two possible contributing factors: reduced access to useful experience and repeated use of misleading experience. First, newly appended entries may compete with earlier entries for retrieval and context space, leaving useful experience stored but unavailable to the agent during execution. Second, self-evaluation may overrate flawed trajectories, and experience induction may produce incorrect rules or omit their conditions of use. Repeated retrieval of these entries could then propagate misleading guidance to later tasks.

Hippo sustains gains over the no-memory baseline on both early and unseen tasks and achieves the highest final performance among the evaluated methods. At the 400-task checkpoint, success rates reach 86% and 82%, respectively, exceeding the strongest memory baseline on each task set by 24 and 22 percentage points. At the same checkpoint, Hippo retains 83 entries, compared with 270 for AWM and 665 for ReasoningBank (Figure 4(b)). These results show sustained gains with fewer stored entries in this task sequence. Also, at the 400-task checkpoint, full Hippo outperforms its append-only variant by 14.0 percentage points on the revisited tasks and 16 percentage points on the unseen tasks (Table 11).

4.3 Exploration Before Memory Updates

More steps. Hajimiri et al. (2026) show that more interaction steps can improve task success without a memory system. We observe the same pattern: with GPT-5.6 Sol, Vanilla-IB gives the agent a larger step budget and improves success rates over No Memory by an average of 0.9 % across four WebArena domains.

More tries. Our experiments reveal a simple insight: *more attempts alone can improve task success*. A method may spend extra computation updating its harness or memory from one trajectory, yet score can lower than the original agent just given more attempts. This motivates an evaluation criterion: a self-evolving method must outperform direct retries under the feedback conditions, and success criterion. Otherwise, the evidence is insufficient to support its claimed effectiveness.

Averaged equally across four WebArena websites, No Memory achieves 64.3% mean per-attempt success and 73.6% success at least once within eight attempts, a difference of 9.3 percentage points. Appendix E.1 provides the metric definitions and normalized gains.

Learning from more tries. This insight motivates Hippo to collect N concurrent trajectories per task using the same policy and retrieved memory. The different paths allow comparisons between successes and failures when both occur. Hippo records surprise-triggered reflections in L_1. A fix for one failure should not automatically become a general rule. We should check whether the problem recurs, specify when the rule applies, and test it on other tasks.

Table 3: **More tries.** For success rates $s_1, s_n \in [0, 1]$, success gain is $\Delta_n = s_n - s_1$. Recovery rate is the fraction of mean per-attempt failures solved by later attempts: $R_n = \frac{s_n - s_1}{1 - s_1}$ $(n = 8)$.

Domain	Single attempt (%)	Any success attempts (%)	Success gain (%)	Recovery rate (%)
Shopping	53.4	65.7	+12.3	+26.4
Shopping Admin	78.9	88.9	+10.0	+47.4
Reddit	56.8	63.6	+6.8	+15.7
GitLab	67.9	76.1	+8.2	+25.5
Mean	64.3	73.6	**+9.3**	**+28.8**

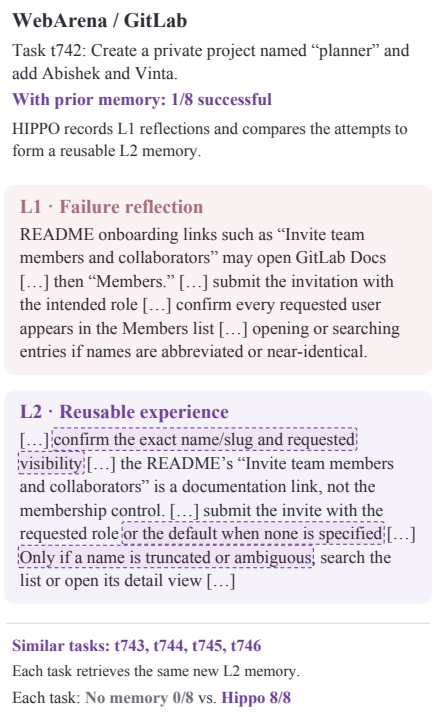

Figure 5: Memory formation and reuse exmaple on WebArena. Excerpts use [. . .] for omissions; dashed boxes mark L2 additions and refinements.

4.4 Number of Rollouts

In our WebArena experiments, tasks receiving memories from majority-success and majority-failure outcomes show average absolute gains in success rate of 23.0% and 13.0%, respectively, compared with 3.3% for the all-failure group. To collect contrasting trajectories within a practical sampling budget, we use eight rollouts per learning task, providing twice as many rollouts as $N = 4$ at half the sampling budget of 16 (Appendix E).

4.5 Case Studies

Zhao et al. (2026) find that replacing experience with meaningless symbols (e.g., `%$#&`) sometimes does not reduce task success, and identified overly general guidance as explanation. Motivated by this, we examine what concrete guidance HIPPO extracts from past attempts and how that guidance is reflected in subsequent execution.

Figure 5 highlights the value of consolidating evidence across attempts. Repeated failures reveal a recurring error, while successful execution provides evidence for a workable alternative. HIPPO brings observations in a reusable procedure with explicit applicability and completion criteria.

The L1/L2 hierarchy thus connects local diagnoses to guidance that remains useful as task-specific details change. The resulting experience specifies both how to act and what evidence is needed to establish completion, with additional inspection conditioned on observable ambiguity. These results illustrate how experience formed from a largely unsuccessful batch can support reliable execution on related tasks. Additional very detailed case studies and failure analyses are provided in Appendix F.

4.6 Ablation Study

We evaluate ablations with GPT-5.6 Sol under the same settings as the main experiments.

Experience hierarchy. Without L_2, the agent retrieves L_1 reflections directly and achieves 66.50% on WebArena and 83.80% on SWE-bench Verified. Without subsequent experience construction and checks against existing memory, these local reflections may retain task-specific details or provide inconsistent guidance on later tasks. Without L_1, we retain the judge and the L_2 engine but omit explicit reflections. The engine builds L_2 experiences directly from raw trajectories and judge feedback. This variant achieves 68.70% and 85.63%, respectively. Explicit reflections may help the engine focus on relevant errors and their context before cross-rollout comparison. Both variants perform worse than full Hippo at 72.90% and 88.50% in Table 4. These results support reflection and subsequent construction of experiences.

Table 4: Ablation study (*w/o*) on WebArena and SWE-bench Verified. *w/o* means without. L_1 denotes episodic reflections and L_2 denotes consolidated experiences.

Method	WebArena (%)	SWE-bench Verified (%)
No Memory	64.20	81.50
w/o L_2	66.50	83.80
w/o L_1	68.70	85.63
Hippo	**72.90**	**88.50**

Comparing multiple attempts. We next examine how the L_2 engine constructs experience (Table 5). Both variants use append-only memory updates. Separate rollout summaries achieve 61.30% success on WebArena, whereas cross-rollout comparison achieves 70.10%. A lesson drawn from one attempt may reflect details specific to that execution. Comparison across attempts provides additional evidence about which lessons to retain and when to apply them. The strategies may also differ in the number and length of stored entries, so this result evaluates the experience construction strategy as a whole.

Table 5: Ablation study on WebArena: experience construction and memory updates.

Method	Success (%)
Separate summaries + append only	61.30
Compare rollouts + append only	70.10
Compare rollouts + repeated rewrite	54.57
Hippo (add and delete)	**72.90**

Memory update strategies. With rollout comparison fixed, Hippo achieves 72.90%, compared with 70.10% for append-only storage and 54.57% for repeated rewriting. Append-only storage leaves earlier guidance unchanged, even when new experience reveals a conflict. Hippo checks when each experience applies and replaces outdated guidance through explicit additions and deletions. The repeated-rewrite variant also permits direct edits and merges of existing entries. Its lower score shows that broader freedom to revise memory does not necessarily improve subsequent performance. Appendix G.3 details these policies.

Contribution of the surprise judge. Replacing the judge with just success/failure labels reduces success rates by 1.8%–8.5% across the five WebArena websites. We observe that answers such as `N/A` or `None` can be marked correct despite an incomplete search or an execution failure. These attempts can then enter memory as successful experience and cause the agent to repeat the same mistakes on later tasks. Map contains the most such cases in our runs and shows the largest decrease, at 8.5%. Detailed results and case studies are provided in Appendix D.

4.7 Efficiency Analysis.

On WebArena, Hippo uses approximately 240–273 memory tokens per injection and 2,800 tokens per rollout receiving memory. With GPT-5.6 Sol, Hippo increases the success rate from 64.2% to 72.9% (+8.7 %) over No Memory. The resource records show that mean calls to the actor model rise from 69.27 to 74.75 (+7.9%), while cost increases from $1.93 to $2.17 per task (+12.4%). Detailed comparisons and accounting scope are provided in Appendix B.

5 Conclusion

Hippo learns from multiple attempts and maintains reusable experience without updating model parameters. Experiments across three task settings show improved performance. In our long-horizon evaluation, Hippo sustains gains with fewer memory entries than the evaluated memory baselines, whose performance eventually falls below No Memory. Our retry analysis also motivates direct retries as a baseline for future evaluations of self-evolving agents. Future work includes reflection within longer tasks and more efficient memory updates. Limitations and future directions are discussed in Appendix H.

AI USE STATEMENT

We used generative AI tools to assist with literature searches and reference checking, manuscript organization and language editing, and feedback on the research methodology, experimental design, and interpretation of results. The authors evaluated the suggestions, checked the references against their original sources, and verified the resulting claims and interpretations against the experimental evidence. The authors take full responsibility for the final manuscript, including all claims, references, and reported results.

REFERENCES

Diogo Almeida. Introducing System One Models & Jev. TypeSafe AI Blog, September 2026. URL https://typesafe.ai/blog/introducing-system-one-models-and-jev. Published September 15, 2026; accessed September 25, 2026.

Patricia W. Cheng and Laura R. Novick. A probabilistic contrast model of causal induction. *Journal of Personality and Social Psychology*, 58(4):545–567, 1990. doi: 10.1037/0022-3514.58.4.545. URL https://doi.org/10.1037/0022-3514.58.4.545.

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.

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

Gordon V. Cormack, Charles L. A. Clarke, and Stefan Büttcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 758–759, 2009. URL https://cormack.uwaterloo.ca/cormacksigir09-rrf.pdf.

Thibault Le Sellier de Chezelles, Maxime Gasse, Alexandre Lacoste, Massimo Caccia, Alexandre Drouin, Léo Boisvert, Megh Thakkar, Tom Marty, Rim Assouel, Sahar Omidi Shayegan, Lawrence Keunho Jang, Xing Han Lù, Ori Yoran, Dehan Kong, Frank F. Xu, Siva Reddy, Graham Neubig, Quentin Cappart, Russ Salakhutdinov, and Nicolas Chapados. The BrowserGym ecosystem for web agent research. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=5298fKGmv3.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-1220. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/5950bf290a1570ea401bf98882128160-Abstract-Datasets_and_Benchmarks.html.

Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Nitin Pasari, Chetan Rane, Karmini Sampath, Maya Krishnan, Srivatsa Kundurthy, Sean Hendryx, Zifan Wang, Vijay Bharadwaj, Jeff Holm, Raja Aluri, Chen Bo Calvin Zhang, Noah Jacobson, Bing Liu, and Brad Kenstler. SWE-Bench Pro: Can AI agents solve long-horizon software engineering tasks? *arXiv preprint arXiv:2509.16941*, 2025. doi: 10.48550/arXiv.2509.16941. URL https://arxiv.org/abs/2509.16941.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. WorkArena: How capable are web agents at solving common knowledge work tasks? In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11642–11662. PMLR, 2024. URL https://proceedings.mlr.press/v235/drouin24a.html.

Bradley Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7 (1):1–26, 1979. doi: 10.1214/aos/1176344552. URL https://doi.org/10.1214/aos/1176344552.

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Qihan Ren, Yiran Wu, Hongru Wang, Han Xiao, Yuhang Zhou, Shaokun Zhang, Jiayi Zhang, Jinyu Xiang, Yixiong Fang, Qiwen Zhao, Dongrui Liu, Cheng Qian, Zhenhailong Wang, Minda Hu, Huazheng Wang, Qingyun Wu, Heng Ji, and Mengdi Wang. 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. URL https://openreview.net/forum?id=CTr3bovS5F.

Sina Hajimiri, Masih Aminbeidokhti, Jose Dolz, Ismail Ben Ayed, Issam H. Laradji, Spandana Gella, and Nicolas Gontier. Are online skill and memory modules always worth their tokens? a budget-constrained study of web agents. *arXiv preprint arXiv:2606.15017*, 2026. URL https://arxiv.org/abs/2606.15017. Accepted to EMNLP 2026.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Dharshan Kumaran and Eleanor A. Maguire. An unexpected sequence of events: Mismatch detection in the human hippocampus. *PLOS Biology*, 4(12):e424, 2006. doi: 10.1371/journal.pbio.0040424. URL https://journals.plos.org/plosbiology/article?id=10.1371/journal.pbio.0040424.

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. doi: 10.1016/j.tics.2016.05.004. URL https://doi.org/10.1016/j.tics.2016.05.004.

John E. Lisman and Anthony A. Grace. The hippocampal-VTA loop: Controlling the entry of information into long-term memory. *Neuron*, 46(5):703–713, 2005. doi: 10.1016/j.neuron.2005.05.002. URL https://doi.org/10.1016/j.neuron.2005.05.002.

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):419–457, 1995. doi: 10.1037/0033-295X.102.3.419. URL https://doi.org/10.1037/0033-295X.102.3.419.

OpenAI. Introducing SWE-bench Verified, 2024. URL https://openai.com/index/introducing-swe-bench-verified/.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. ReasoningBank: Scaling agent self-evolving with reasoning memory. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=jL7fwchScm.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023. doi: 10.48550/arXiv.2310.08560. URL https://arxiv.org/abs/2310.08560.

Jiayi Pan, Yichi Zhang, Nicholas Tomlin, Yifei Zhou, Sergey Levine, and Alane Suhr. Autonomous evaluation and refinement of digital agents. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=NPAQ6FKSmK.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pp. 1–22, 2023. doi: 10.1145/3586183.3606763. URL https://doi.org/10.1145/3586183.3606763.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009. doi: 10.1561/1500000019.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. Is ChatGPT good at search? investigating large language models as re-ranking agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 14918–14937. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.923. URL https://aclanthology.org/2023.emnlp-main.923/.

vLLM Project. Automatic prefix caching. vLLM documentation, n.d. URL https://docs.vllm.ai/en/latest/features/automatic_prefix_caching/. Accessed September 25, 2026.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63897–63911. PMLR, 2025. URL https://proceedings.mlr.press/v267/wang25bx.html.

Matthew A. Wilson and Bruce L. McNaughton. Reactivation of hippocampal ensemble memories during sleep. *Science*, 265(5172):676–679, 1994. doi: 10.1126/science.8036517. URL https://doi.org/10.1126/science.8036517.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. C-Pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024. doi: 10.1145/3626772.3657878. URL https://arxiv.org/abs/2309.07597v5.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-MEM: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-0593. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/19909c36f51abc4856b4560aff3d36d6-Abstract-Conference.html.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schütze, Volker Tresp, and Yunpu Ma. Memory-R1: Enhancing large language model agents to manage and utilize memories via reinforcement learning, 2025. URL https://arxiv.org/abs/2508.19828v5. Version 5, revised January 14, 2026.

Ye Yu, Xiaopeng Yuan, Haibo Jin, Heming Liu, Yaoning Yu, and Haohan Wang. Do self-evolving agents forget? capability degradation and preservation in lifelong llm agent adaptation. *arXiv preprint arXiv:2605.09315*, 2026. doi: 10.48550/arXiv.2605.09315. URL https://arxiv.org/abs/2605.09315.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin gödel machine: Open-ended evolution of self-improving agents. In *International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=pUpzQZTvGY.

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

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

Weixiang Zhao, Yingshuo Wang, Yichen Zhang, Yang Deng, Yanyan Zhao, Wanxiang Che, Bing Qin, and Ting Liu. Large language model agents are not always faithful self-evolvers. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2601.22436.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 62557–62583, 2024. doi: 10.52202/079017-2000. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/724be4472168f31ba1c9ac630f15dec8-Abstract-Conference.html.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=oKn9c6ytLx.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Jinhua Zhao, Bryan Kian Hsiang Low, and Paul Pu Liang. MEM1: Learning to synergize memory and reasoning for efficient long-horizon agents, 2025. URL https://arxiv.org/abs/2506.15841v2. Version 2, revised July 17, 2025.

A Appendix

A.1 Interaction steps.

We analyze Hippo's rollout lengths to examine how much of the 30-step interaction budget is used in practice (Table 6). On Shopping, Shopping Admin, GitLab, and Reddit, completed rollouts use 8.12–9.40 steps on average, with 90th-percentile lengths of 15–20 steps. Map has longer rollouts, with a mean of 13.39 steps and a 90th percentile at the 30-step limit. Across websites, 2.8–10.3% of completed rollouts reach this limit; most therefore terminate before exhausting their step budget.

Table 6: Interaction steps of Hippo on WebArena.

Website	Completed rollouts	Mean steps	Median steps	P90 steps	Reached 30 steps
Shopping	1,492	8.12	5	20	100 (6.7%)
Shopping Admin	1,430	8.85	7	17	94 (6.6%)
GitLab	1,430	9.40	8	19	40 (2.8%)
Reddit	791	8.76	7	15	28 (3.5%)
Map	836	13.39	12	30	86 (10.3%)

Note. The interaction limit is 30 steps per rollout. Statistics cover completed rollouts regardless of success, including those that reach the limit; crashes are excluded. P90 denotes the 90th percentile. Percentages use the completed-rollout count for each website.

B Efficiency Analysis

We examine memory input, model calls, and an API cost proxy on WebArena with GPT-5.6 Sol and eight rollouts per task. Per-task statistics use 763 records for No Memory and 761 for Hippo, with two Hippo task-level error cases excluded from the task count. Memory input statistics cover 6,064 recorded Hippo rollouts, of which 3,750 receive memory.

Memory input. Mean memory input ranges from 240 to 273 token equivalents per injected step across websites (Table 7). Repeated use within a rollout increases cumulative input to an overall mean of 2.8k token equivalents per injected rollout; site-level p90 values range from 3.80k to 8.42k. We use four characters per token equivalent and count each injection separately. The resulting totals measure memory text supplied during execution, rather than stored memory size or total API token usage.

Table 7: Memory input in Hippo.

	Rollouts		Tokens per step	Per rollout (k tokens)			Total
Website	Injected	Recorded	Mean	Mean	Median	p90	(M)
Shopping	1,072	1,496	262	2.39	1.49	6.86	2.56
Shopping Admin	696	1,455	273	2.98	2.18	7.77	2.07
GitLab	720	1,438	249	2.52	2.12	5.00	1.81
Reddit	530	839	240	2.30	2.02	3.80	1.22
Map	732	836	268	3.86	3.28	8.42	2.83
Overall	**3,750**	**6,064**	**260**	**2.80**	—	—	**10.5**

Note. Token equivalents equal character counts divided by four. Step and rollout statistics are conditional on memory injection. Cumulative values include repeated injections; pooled median and p90 are unavailable.

Execution and memory operations. Memory operations account for 4.1 of Hippo's 78.8 model calls per task; the remaining 74.7 calls execute the agent. Relative to the 69.3 calls per task used by No Memory, the combined call count increases by 13.8%. Table 8 separates L1 reflection, L2

writing, and gating, and reports L1 invocations alongside the reflections produced. These counts distinguish the work spent processing memory from the number of entries written.

Table 8: Memory operations in Hippo.

	L1 reflection		L2 writing	Gating	Total
Website	Calls	Written	Calls	Calls	Calls
Shopping	630	560	103	186	919
Shopping Admin	500	454	84	179	763
GitLab	360	314	65	178	603
Reddit	153	142	43	104	300
Map	332	303	74	108	514
Overall	**1,975**	**1,773**	**369**	**755**	**3,099**

Note. L1 invocations and produced reflections are separate quantities. Total calls sum L1, L2, and gating.

API cost proxy. We also report a monetary cost proxy because calls can differ in their input and output lengths. It uses USD figures for both methods and covers task execution, trajectory assessment, and memory operations, including injected memory text (Table 9). The proxy averages $2.17 per task for Hippo and $1.93 for No Memory. But, the relative increase is 12.4%.

Table 9: API cost proxy in USD.

	No Memory			Hippo		
Website	Tasks	Total ($)	$/task	Tasks	Total ($)	$/task
Shopping	187	251.15	1.34	187	335.50	1.79
Shopping Admin	182	656.82	3.61	182	669.98	3.68
GitLab	180	302.38	1.68	180	341.73	1.90
Reddit	106	150.42	1.42	105	166.50	1.59
Map	108	111.24	1.03	107	135.78	1.27
Overall	**763**	**1,472.02**	**1.93**	**761**	**1,649.50**	**2.17**

Note. The proxy includes all related model calls. Overall means pool the cost proxy and task counts across websites.

B.1 Memory Retrieval and Selection

We evaluate memory selection on 752 recorded WebArena queries (Zhou et al., 2024), using the same-site L_2 experiences available at each query's retrieval time. The baselines are replayed on these historical banks and compared with Hippo's recorded selections, without rerunning the downstream agent. An experience counts as a match if its source task shares the query's intent template. Selection rate is measured on the 363 queries with a match available; correct abstention is measured on the remaining 389.

We compare Hippo with BM25 (Robertson & Zaragoza, 2009), dense top-1 retrieval with BGE (Xiao et al., 2024), hybrid sparse–dense ranking via reciprocal rank fusion (Cormack et al., 2009), and similarity-threshold policies. As shown in Table 10, Hippo achieves the highest selection rate among these policies: 75.2%, compared with 66.4% for BM25 with experience content. The paired gain is 8.8 percentage points, with a 95% task-bootstrap confidence interval (Efron, 1979) of [4.1, 13.5]. When no match is available, Hippo abstains in 62.0% of queries. A cosine threshold of 0.65 raises correct abstention to 72.2%, but reduces selection to 33.3%. We also evaluate direct LLM selection from the full historical memory bank. Its selection rate is 47.8%, compared with 75.2% for Hippo. This result supports a short candidate list before the LLM selects an experience.

Retrieving several candidates gives the selector access to matches below rank one. Among the 363 queries with a match available, dense top-1 selects one in 217 cases, while the top-5 contains one in 300 (Figure 6). The LLM selects a match in 273 of those 300 cases, a conditional selection rate

Table 10: Memory selection and abstention.

Policy	Selection ↑	Correct abstention ↑
BM25	62.8	1.3
BM25 (+content)	66.4	0.8
Dense top-1	59.8	0.0
Hybrid (RRF)	65.3	0.0
Dense + $\tau = 0.65$	33.3	72.2
Dense + $\tau = 0.75$	7.4	99.5
LLM-only (full memory bank)	47.8	n/a
Hippo: cosine top-5 + LLM	**75.2**	62.0

Note. Relevance uses the same-template criterion. All values are percentages. BM25 (+content) also indexes the experience body; all dense policies use bge-small.

of 91.0%. The resulting selection rate is 273/363, or 75.2%; the remaining 90 queries comprise 63 without a match in the top-5 and 27 with a match that the LLM does not select.

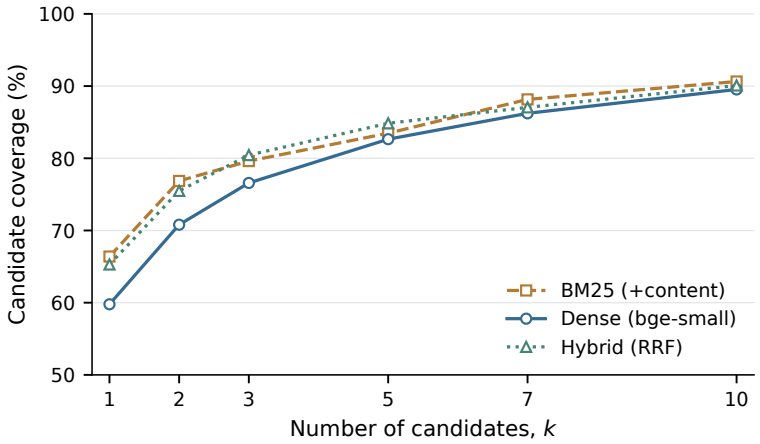

Figure 6: Candidate coverage on the 363 queries with a same-template experience available. Coverage is the fraction of queries whose top-k candidates contain a match. The curves measure candidate retrieval. Among shortlist-based methods, only cosine top-5 was evaluated with the LLM selector. Direct selection from the full memory bank was evaluated separately.

Template agreement is a relevance proxy, and Hippo uses an additional LLM call. The comparison therefore does not establish downstream gains at equal cost. Other candidate generators and shortlist sizes were not tested with the same LLM, so their relative effectiveness remains open. Within this scope, the results support cosine retrieval followed by LLM selection for identifying same-template experiences.

C Experimental Details

Model use. Each Hippo run uses one model for task execution, trajectory assessment, and memory updates. Model parameters remain fixed throughout the run.

SWE bench Verified. We use all 500 tasks in the `test` split of `princeton-nlp/SWE-bench_Verified`. Each attempt runs in a separate task container. We use Epoch AI arm64 images where available. Supplementary images or separate runs cover the remaining tasks. For the main results, we evaluate all eight patches per task with the official `swebench` 4.1.0 harness. We average the eight binary outcomes within each task, then across all 500 tasks. Each Hippo run starts with an empty memory bank. The memory engine updates the bank only after all eight attempts for a task have finished, so experiences from that task

become available only to subsequent tasks. All methods process the tasks in task-ID order, with the same order across the three runs.

WebArena protocol. The main results use eight rollouts per task. All methods and ablated variants use a maximum of 30 interaction steps per rollout, except Vanilla-IB, which uses 50. Tasks follow task-ID order, and all Hippo rollouts for a task use the same memory snapshot. Hippo completes the rollouts and updates memory before the next task begins. BrowserGym (Drouin et al., 2024; de Chezelles et al., 2025) calls the WebArena evaluator at each environment step, and the last returned reward defines each rollout's outcome. We average success over the eight rollouts within each task, then across tasks.

Mind2Web. We evaluate Mind2Web offline under teacher forcing, using gold action histories and candidate sets constructed from gold element labels. For each attempt, we average step success within the task. For the main results, we then average the eight attempt scores for each task, followed by an average across tasks. At each step, the page and execution context are reset, while persistent memory is retained. Memory also carries over across the three benchmark splits. AWM, ReasoningBank, and Hippo use the same split order and memory reset policy.

Long-horizon task construction. The task collection for the long-horizon experiment includes coding tasks selected from SWE-bench Pro and processed for evaluation. All methods use the same task collection and follow the same task order.

WebArena infrastructure. We host five WebArena websites on eight AWS EC2 `m6i.xlarge` instances. Each instance has 4 vCPUs, 16 GB of memory, and a 1000 GiB `gp3` root volume with 3000 IOPS. The Shopping, Shopping Admin, Reddit, and GitLab images occupy about 424 GB in total. Concurrent rollouts use separate deployments. Before each comparison condition, we restore the website containers from the baseline images.

Infrastructure costs. At the quoted deployment rates, the eight EC2 instances cost approximately USD 1.54 per hour. Their eight storage volumes cost approximately USD 640 per month. With continuous operation and a 30-day month, this corresponds to about USD 58.3 per day for compute and storage. These estimates cover WebArena compute and storage and exclude model API calls and other charges.

C.1 Long-horizon comparison with append-only memory

We compare full Hippo with its append-only variant at the 400-task checkpoint. The revisited set contains the first 50 tasks, while the unseen set contains 50 fixed tasks. Evaluation does not update the memory used for subsequent learning. As shown in Table 11, full Hippo achieves higher final success on both sets. The comparison supports the value of memory updates beyond the addition of new experience.

C.2 Success rates after learning tasks

Table 11: Success rates (%) after 400 learning tasks, evaluated on the first 50 tasks and a fixed set of 50 unseen tasks.

Method	Revisited	Unseen
Append-only Hippo	72.0	66.0
Full Hippo	**86.0**	**82.0**

D Judge Reliability and Contribution to Learning

D.1 Why outcome labels are not enough

Our WebArena runs reveal that an accepted response can come from an incomplete attempt. Among 412 accepted traces flagged by the judge as fortuitous successes, 193 (46.8%) have N/A-containing references, and 192 of these have no final response (Table 12). These counts describe

judge-flagged traces, not all WebArena tasks. A failed search or an early stop can thus receive the same score as a careful check that finds no matching result. If the accepted trace becomes a successful example for memory construction, the agent may learn to repeat an unsupported decision.

Hippo uses the judge to assess the evidence behind accepted responses and to diagnose failed attempts. The official score remains fixed: an incorrect attempt is labeled `failure`, while an accepted attempt is labeled `genuine` or `fluke`. When the evidence for an accepted attempt is ambiguous, the judge defaults to `genuine`. These labels guide reflection and experience construction without changing benchmark scores.

Table 12: Composition of the recorded WebArena pool.

Website	Accepted	Flagged	`N/A` reference	No answer	Both
Shopping	1,698	99	83	99	83
Shopping Admin	1,859	44	0	32	0
GitLab	2,103	103	31	100	30
Reddit	1,278	26	23	26	23
Map	999	140	56	88	56
Total	7,937	412	193	345	192

Note. Accepted counts all traces accepted by the evaluator in the full recorded pool. Flagged counts those subsequently labeled fortuitous successes by the judge. The last three columns are counts within the flagged set.

D.2 Feedback available during memory construction

WebArena Within Hippo, the judge receives the reference answer and official outcome along with the task, final response, and trace. The L_1 and L_2 writers receive its labels and diagnoses, but no separate reference-answer field (Table 13). A diagnosis can still contain reference-derived values or assessment criteria. The actor receives only selected past experience in addition to its normal task inputs; it has no direct access to the current task's reference, reward, or judge reason.

Table 13: Feedback and experience available to each Hippo component.

Component	Feedback and experience
Judge	Official outcome, reference, and observed trace
L_1 writer	Judge label and diagnosis; individual trace
L_2 writer	Outcomes and diagnoses; L_1 or genuine-success traces
Actor	Selected past experience; no direct current-task feedback

In our WebArena comparison, AWM and ReasoningBank also receive official outcomes and reference answers during memory construction. We enable MaTTS for ReasoningBank. The three methods therefore have access to reference information, although they use it differently. This is an adaptation of their original online settings, which use model-based outcome judgments (Wang et al., 2025; Ouyang et al., 2026).

The original evaluators also serve a different purpose. AWM adopts the evaluator of Pan et al. (2024), while ReasoningBank uses its agent backbone to predict success or failure. ReasoningBank reports 72.7% accuracy for Gemini-2.5-flash on WebArena-Shopping (Ouyang et al., 2026). Hippo instead takes the official result as given and assesses the trace or diagnoses the failure.

Mind2Web. We evaluate Mind2Web as an offline stream under teacher forcing, using gold action histories and candidate sets constructed from gold element labels. At each step, the page and execution context are reset, while Hippo's persistent L_2 memory is retained. The memory bank also carries over across the Cross-Task, Cross-Website, and Cross-Domain splits. AWM, ReasoningBank, and Hippo use the same split order and memory reset policy. The reported scores therefore describe adaptation along a shared memory stream. The judge is not told which candidate is correct. For each attempt, we average step success within the task. We then average the eight attempt scores for each task, followed by an average across tasks.

SWE-bench Verified. The judge is label-free. It receives the issue, execution trace, and candidate patch, without the reference patch or official outcome label. It returns two binary fields, `success` and `verified`, for memory construction. The reported resolution rate is computed separately with the official SWE-bench harness.

D.3 Effect of the judge

We replace the judge with official success/failure labels. This removes both the genuine/fluke distinction and the textual diagnoses, including those for failed attempts. Success rates fall on all five WebArena websites, by 1.8–8.5 percentage points (Table 14). The comparison measures the combined contribution of trajectory assessment and diagnostic feedback.

Map contains the most N/A or None cases in our runs and has the largest decrease, at 8.5 percentage points. Such cases make the distinction between a verified absence and an incomplete attempt especially relevant. The case in Section D.6 illustrates this distinction; Section F.3 shows how failure diagnoses can support experience construction.

Table 14: Effect of removing the judge.

Website	Decrease (pp)
Shopping	3.5
Shopping Admin	1.8
GitLab	3.5
Reddit	1.8
Map	8.5

Note. Success-rate decrease relative to full Hippo, in percentage points (pp).

D.4 Agreement with human annotations

Two human annotators assess 200 evaluator-accepted traces, with 50 traces from each of four actor models. For each of the five websites, the audit includes 20 randomly sampled traces labeled as genuine successes and 20 labeled as fortuitous successes by the judge. The annotators agree on 195 of the 200 traces (97.5%; Cohen's $\kappa = 0.95$). After excluding five disagreements and one inconclusive case, the judge agrees with the human labels on 183 of the remaining 194 traces (94.3%), with Cohen's $\kappa = 0.89$. Among the 94 retained traces flagged as fortuitous successes, humans confirm 86 (91.5%). Table 15 reports the results by website. Agreement varies across sites: the judge matches all retained human labels on Shopping and Reddit, while agreement on Map is 84.6% ($\kappa = 0.69$). All metrics describe the stratified audit sample, with judge–human comparisons restricted to conclusive human agreement.

Agreement measures. We report raw agreement and Cohen's κ (Cohen, 1960) separately for the two human annotators and for the judge against conclusive human consensus. For the binary judge–human comparison, let $n = \mathrm{TP} + \mathrm{FP} + \mathrm{FN} + \mathrm{TN}$. The observed and chance-expected agreements are

$$p_o = \frac{\mathrm{TP} + \mathrm{TN}}{n}, \qquad p_e = \frac{(\mathrm{TP} + \mathrm{FP})(\mathrm{TP} + \mathrm{FN}) + (\mathrm{FN} + \mathrm{TN})(\mathrm{FP} + \mathrm{TN})}{n^2}, \tag{5}$$

and $\kappa = (p_o - p_e)/(1 - p_e)$. Using the pooled counts in Table 16, $p_o = 183/194$, $p_e = 18{,}866/37{,}636$, and $\kappa = 0.8863$, reported as 0.89. The pooled coefficient is computed from pooled counts rather than an average of site-level coefficients. Flag precision is $\mathrm{TP}/(\mathrm{TP} + \mathrm{FP})$ within the retained audit sample. These statistics do not correct for the stratified sampling design or the exclusion of unresolved labels.

D.5 Repeatability on the same input

Repeated judging produces consistent labels on 24 of 25 checked traces (96.0%). This check measures consistency across repeated evaluations of the same input, separately from agreement with human labels.

Table 15: Agreement in the human audit of judge labels.

	Human–human		Judge–human			Flag precision
Website	Agree (%)	κ	n	Agree (%)	κ	(%)
Shopping	100.0	1.00	40	100.0	1.00	100.0
Shopping Admin	90.0	0.79	36	91.7	0.83	81.3
GitLab	100.0	1.00	39	94.9	0.90	94.7
Reddit	100.0	1.00	40	100.0	1.00	100.0
Map	97.5	0.95	39	84.6	0.69	78.9
Pooled	**97.5**	**0.95**	**194**	**94.3**	**0.89**	**91.5**

Note. Each website contributes 40 traces, split equally between the judge's two labels. Human–human metrics use all 200 traces; judge–human metrics use the $n = 194$ traces with matching, conclusive human labels. Flag precision is the percentage of retained judge-flagged traces confirmed by humans. Metrics are unweighted descriptions of the stratified sample; pooled κ is computed from pooled labels.

Table 16: Human audit counts for evaluator-accepted WebArena traces.

Website	Audited	Retained	TP	FP	FN	TN
Shopping	40	40	20	0	0	20
Shopping Admin	40	36	13	3	0	20
GitLab	40	39	18	1	1	19
Reddit	40	40	20	0	0	20
Map	40	39	15	4	2	18
Total	200	194	86	8	3	97

Note. Each website contributes 20 judge-genuine and 20 judge-flagged traces. Retained traces have matching, conclusive human labels. Fortuitous success is the positive class: TP and FP count retained judge-flagged traces, while FN and TN count retained judge-genuine traces. Counts describe the stratified audit sample.

D.6 Cases of incomplete verification

D.6.1 An accepted response without completed verification

In the first case, the reference is N/A. The agent takes one action and returns no final answer, yet the fuzzy-match evaluator accepts the response. The trace indicates that a required check is still unfinished, so the judge labels the attempt as `fluke`. The record has no task identifier.

A1 What the evaluator accepted

The agent inspected a project list but did not complete its check of the user's contribution activity.

Field	Recorded value
Reference	`N/A` (fuzzy match)
Final response	""
Actions	1
Official result	Correct
Judge label	`fluke`

A2 Why the judge flagged it

Judge reason (verbatim)
"The agent produced no final answer. Although it inspected the project list and noticed no displayed repository exceeded 100 stars, it explicitly concluded that the profile contribution activity still needed inspection and never completed that verification. The empty response only happened to align with the N/A reference."

The `fluke` label marks the attempt for reflection. The record contains no experience generated from this trace.

D.6.2 How an unsupported procedure can enter memory

An earlier GitLab experiment illustrates how an accepted outcome can enter memory without a verified procedure behind it. Task t168 asks for repositories to which the user contributed that have more than 100 stars; its reference answer is `N/A`. In the earlier implementation, the experience writer could access the reference answer and produced a lesson that advised returning `N/A` if no repository exceeded the threshold. Three related tasks (t169–t171) subsequently retrieved this lesson and returned `N/A`, although their reference answers named repositories. Each task failed in all eight attempts.

A later implementation both excluded fortuitous successes from the contrast used for induction and withheld the raw reference-answer field from the writer. It produced an experience that instructed the agent to open Projects, select Yours, sort by Most stars, and verify the star counts. The same three related tasks each succeeded in all eight attempts; an additional related task, t172, also succeeded in all eight attempts. This historical comparison illustrates how different inputs to experience induction can accompany different stored lessons and subsequent behavior. Because both trace filtering and direct access to the reference-answer field changed, it does not isolate the causal contribution of the surprise judge or evaluate the current implementation.

E Rollout Outcomes and Budget

Setup and measures. We analyze 761 WebArena tasks with gpt-5.6-sol at $N = 8$. Of these tasks, 471 received experience and 290 did not. We group the 471 tasks by the outcomes of the rollouts used to produce their experience. The balanced group has equal counts of successes and failures. Both majority groups include at least one success and one failure. Let n_g be the task count of group g and Δ_g its mean score gain. All groups use the same reference condition. The group's contribution C_g to the overall score is

$$\underbrace{C_g}_{\text{overall contribution}} = \underbrace{\frac{n_g}{M}}_{\text{task coverage}} \times \underbrace{\Delta_g}_{\text{within-group gain}}, \qquad M = 761, \tag{6}$$

where C_g and Δ_g are in percentage points (pp).

Table 17: Gains by experience source at $N = 8$.

Experience source	Tasks	Δ_g (pp)	C_g (pp)
All failure	183	+3.3	+0.80
Majority failure	140	+13.0	+2.39
Balanced	24	+55.8	+1.76
Majority success	124	+23.0	+3.75

Note. Both gain columns are in percentage points. Group means are rounded to one decimal place.

Gain and coverage. The all-failure group has the smallest mean gain (Table 17). The balanced group has the largest, but covers only 24 tasks and adds 1.76 pp to the overall score. The two majority groups have smaller gains per task, but cover 264 tasks and add 6.14 pp in total.

Rollout budget. With $N = 1$, we cannot compare success and failure within a task. With $N = 2$, such a comparison requires one success and one failure. The larger choices, $N = 4, 8, 16$, also allow unequal counts. We choose $N = 8$ because it allows more attempts than $N = 4$ and needs half as many as $N = 16$. These results do not show which N gives the best performance or the largest gain per token.

E.1 Metrics for Repeated Attempts

We compare two success measures for the No Memory agent with GPT-5.6 Sol on four WebArena websites: Shopping, Shopping Admin, Reddit, and GitLab. For website d, let p_d denote the mean

per-attempt success rate reported in the main table. Let $a_{8,d}$ denote the fraction of tasks solved at least once within eight attempts, without memory updates.

With both rates expressed as probabilities, we compute the absolute gain Δ_d in percentage points and the normalized gain G_d as

$$\Delta_d = 100(a_{8,d} - p_d), \qquad G_d = 100\frac{a_{8,d} - p_d}{1 - p_d}. \tag{7}$$

For $p_d < 1$, G_d expresses the gain as a percentage of the mean per-attempt failure rate.

We compute each metric separately for each website and then average the four website-level values equally. The resulting mean per-attempt and at-least-once success rates are 64.3% and 73.6%, respectively. The mean absolute gain is 9.3 percentage points. The website-level normalized gains are 26.4%, 47.4%, 15.7%, and 25.5%, giving a mean of 28.8%.

The normalized gain compares the two success measures. It does not track which tasks failed on the first attempt and succeeded on a later attempt.

F Qualitative Analysis of Learned Experiences

We examine five cases that illustrate how Hippo turns recorded reflections and trajectory evidence into experiences that can be retrieved on later tasks. The cases cover conflicting recommendations, transfer between task templates, learning when all source attempts fail, the consolidation of an execution procedure, and reuse across capacity requirements. For Cases 1–4, each box presents selected L1 passages together with the complete title, applicability description, and procedural content of the resulting L2. The surrounding text explains the source evidence, the aspect of the method illustrated, and the observed behavior on later tasks. All counts are successful attempts out of eight within the corresponding run. Source results precede formation of the displayed L2 and should not be attributed to that new experience.

Presentation of memory records. Memory wording is retained apart from sentence punctuation and menu-path notation. Semicolons are replaced with sentence breaks, and menu paths use slashes. Bracketed ellipses indicate omissions from L1. For Cases 1–4, L2 applicability and procedure fields are shown in full.

F.1 Case 1. Reconciling reflections through explicit applicability

Task t62 asks which customer has completed the most orders over the full history. HIPPO retrieves no prior experience on this task and succeeds in 4/8 attempts, compared with 6/8 without memory. The trajectories expose a distinction between obtaining the right name and using a data source that respects the question. Some successful attempts aggregate the general Customer Order Count report, while another uses orders restricted to Complete. The recorded L1 reflections also differ. Some recommend aggregating the report across years, whereas another explicitly requires status-filtered orders. HIPPO subsequently forms the L2 shown below.

Experience 1. WebArena / Shopping Admin (formed after t62)

L1 reflection (excerpt): Aggregate the full Order Count report by customer
Open REPORTS, then Customers / Order Count. Set a realistic full-history range, choose Year in "Show By" to avoid oversized daily reports, and select Refresh. Read every populated interval and sum the Orders value for every repeated customer name across all rows and years, then compare the resulting customer totals and report every tie. [. . .]
L1 reflection (excerpt): Count completed orders from Sales Orders
Open Sales / Orders, expose the full historical grid, filter Status to Complete/Completed, and inspect every page or export all matching rows before grouping by the exact full Customer Name shown. [. . .]

L2 experience: Use the status-filtered grid and aggregate every occurrence
Applicability. Applies to historical customer rankings, especially when the question restricts orders by status. The Reports / Customers / Order Count report is suitable only when all statuses qualify or its scope is independently verified.
Procedure. For status-specific counts, use Sales / Orders, open Filters, apply the required Status, and ensure every matching record is visible across all pages or an export. Do not trust Order Count as a

Experience 1. WebArena / Shopping Admin (formed after t62) (continued)

look-alike because it has no status filter. Group every exact full customer name consistently across the complete result—including repeated rows and intervals—compare final totals, include every tie, distinguish near-identical names, and open View if a grid value is abbreviated before answering only from visibly confirmed data.

What this case illustrates. HIPPO consolidates differing L1 recommendations into guidance with explicit applicability. The L2 preserves aggregation and tie handling while requiring status-filtered data for status-specific questions. This illustrates how consolidation retains task constraints across competing recommendations. The source judge also accepts correct answers from the general report, so the case supports conditional guidance rather than reliable detection of spurious success.

Reuse and observed effect. On t63, which asks for the second-most completed orders, HIPPO retrieves the new L2 and achieves 8/8 successes versus 2/8 without memory. A representative attempt checks all 153 Complete orders and returns Adam Garcia, Michael Nguyen, and Sarah Miller, tied at eight orders each. Both tasks share template 276.

F.2 Case 2. Transferring a verification procedure across task templates

Task t126 asks for the price range of Canon photo printers and belongs to template 159. With an older price-range memory, HIPPO succeeds in 7/8 attempts, compared with 8/8 without memory. The single failed HIPPO attempt already finds the correct endpoints but does not submit a final answer. Successful source attempts also demonstrate how to exclude irrelevant products. The resulting experience preserves the working search and verification procedure together with the reminder to stop and answer once the endpoints are established. Thus, the batch contains useful information even though most source attempts already succeed.

Experience 2. WebArena / Shopping (formed after t126)

L1 reflection (excerpt): Verify the complete range, then answer immediately
Open "Advanced Search," constrain "Product Name" with the most reliable brand or identifying term, increase "Show" to the largest page size, sort by "Product Name," and inspect every result page. Treat cables, parts, supplies, cases, and near-identical nonqualifying variants as look-alike traps. [...] once both visible endpoints are verified across the complete set, stop navigating and explicitly return the price range.

L2 experience: Verify the complete product range, then answer
Applicability. Use when a shopping task asks for a price range or other extrema across products. Do not use a broad catalog scan when a single product detail page already answers the question.
Procedure. Open "Advanced Search," constrain "Product Name" with the brand or identifying product term, use the largest "Show" option, and inspect every results page or sort by price. Exclude accessories, parts, supplies, and products that merely mention the brand. Read the full names and prices directly, opening product details if the grid abbreviates them, account for every qualifying or tied row, and once both visible endpoints are verified, stop navigating and explicitly provide the requested range—do not omit the final answer.

What this case illustrates. The L2 retains a product-verification procedure and the single failed attempt's reminder to submit an answer once both endpoints are confirmed. Its scope covers product ranges across brands. Reuse from template 159 to template 370 illustrates procedural transfer within one shopping site, even though the source failure concerns answer submission and the target failures concern product qualification.

Reuse and observed effect. Tasks t226 and t228 both retrieve this L2. HIPPO achieves 8/8 on each, compared with 0/8 and 1/8 without memory, respectively. On t226, a representative attempt excludes a Polk audio bundle that only includes an Amazon Basics cable and reports $5.49–$375.19. On t228, the runs return the Sephora range of $18.18–$94.99.

F.3 Case 3. Forming useful experience without a successful source attempt

Task t141 asks how much was spent on food-related shopping during March 2023. Both arms achieve 0/8, and no memory is retrieved for the HIPPO attempts. The recorded failure diagnoses identify omitted category-related items and missing attributable shipping. L1 reflections preserve these corrections, and HIPPO forms an L2 that combines complete order-history inspection, category coverage, and attributable charges. There is no successful source demonstration in this batch. The experience instead draws on the information supplied by failed attempts and their diagnoses.

Experience 3. WebArena / Shopping (formed after t141)

L1 reflection (excerpt): Use full order details and include attributable costs
From "My Account," open "My Orders" rather than relying on the incomplete "Recent Orders" table, review every relevant page and date row, then use each row's "View Order" link to inspect "Items Ordered" and "Shipping & Handling." Enumerate every qualifying product—including clearly related supplies, decorations, or accessories rather than only consumables—read its full displayed name, variant, quantity, and subtotal, and include the shipping attributable to each qualifying item when the order details establish a per-item charge. [. . .]

L2 experience: Audit complete orders and allocate related shipping
Applicability. Use for category-spend questions based on purchase history, especially when orders mix qualifying and unrelated merchandise. Do not rely on the limited "Recent Orders" panel or interpret a category as consumables only.
Procedure. Open "My Account" / "My Orders," review all pages for the requested period, and select "View Order" on every qualifying row. In "Items Ordered," enumerate every visibly category-related product—including accessories, decorations, and supplies—using full names and displayed subtotals, then add each item's attributable share of "Shipping & Handling" based on the visible per-item or per-unit pattern and reconcile the result with the order totals.

What this case illustrates. HIPPO forms reusable guidance from an entirely failed batch by combining category coverage and shipping checks. The diagnoses provide reference-informed corrections, so the learning signal exceeds binary reward. Transfer remains conditional: t142 retrieves the same experience but achieves 0/8, as does the baseline. Its equal shipping allocation is rejected, showing that the experience's allocation assumptions still require verification.

Reuse and observed effect. Task `t143` asks about home-decoration spending on January 29, 2023. The selector chooses this experience from five candidates (index 0; cosine similarity 0.585). All eight Hippo attempts receive the same optional hint, recorded as 746 characters. The instruction tells the actor to use it only when it clearly fits the current task. All eight no-memory attempts return $260.69, while all eight Hippo attempts return $265.69, giving 0/8 versus 8/8 success. A separate diagnosis for no-memory rollout 0 on `t143` attributes the $5 difference to the item's share of a $10 shipping charge. That later diagnosis is not the source of the earlier `t141` experience. Section G provides the recorded prompt excerpts for these roles.

F.4 Case 4. Consolidating an execution procedure and its completion checks

Task t501 asks the agent to mark all Taurus Elements Shell products and variants out of stock. HIPPO retrieves an older experience about verifying salable inventory and succeeds in 1/8 attempts, compared with 0/8 without memory. Failures occur at several stages. Some attempts cannot select all matching rows, others fail to perform the required stock update, and others stop when an asynchronous operation has only been submitted. Several attempts also mistake a pre-existing success notification for confirmation of the current task. The L1 reflections record corrections at these different stages, and the new L2 assembles them into the procedure below.

Experience 4. WebArena / Shopping Admin (formed after t501)

L1 reflection (excerpt): Verified bulk stock update
[. . .] Select the complete filtered set using the header Options menu's Select All when the adjacent checkbox is obstructed, confirm the selected count, then use Actions to open the inventory/source-stock update, set stock status to out of stock, and save or confirm. [. . .]
L1 reflection (excerpt): Confirm queued bulk stock updates
[. . .] "Message is added to queue" confirms submission only: wait for a new job-specific success notification, ignoring older System Messages, then refresh and open matching product detail views as needed to verify every Stock Availability field visibly shows the requested value rather than inferring it from quantity, salable quantity, or product status.

L2 experience: Select all, update Stock Availability, and respect the queue
Applicability. Bulk-changing Stock Availability for every matching product and variant in Catalog / Products. Not changing product enablement status.
Procedure. Filter the Products grid, inspect every matching parent and variant, use Options / Select All, confirm the selected count, then choose Actions / Update attributes / Advanced Inventory, check Change beside Stock Availability, choose the requested value, and Save. "Change status" and the grid's Status column control enablement, not inventory. If "Message is added to queue" appears, wait for a new operation-specific success notice before verification, ignoring pre-existing System Messages. Only then refresh and open product details as needed to read the actual Stock Status, rather than inferring it from quantity or salable quantity.

What this case illustrates. Consolidation organizes L1 corrections into a procedure that distinguishes inventory from product enablement and submission from completion. Execution remains imperfect: one failed t502 attempt selects all 16 products and submits the update, then reports completion while it is still queued. Some successful episodes terminate at saving without a later queue check. The case shows useful procedural guidance alongside incomplete compliance.

Reuse and observed effect. Tasks t502 and t504 retrieve the new L2 for Gobi HeatTec Tee and Selene Yoga Hoodie products. HIPPO achieves 4/8 and 6/8, compared with 0/8 and 2/8 without memory, respectively. Successful attempts select the matching products and use Advanced Inventory to set Stock Availability to Out of Stock.

F.5 Case 5. Reuse across capacity requirements

Task `t158` asks for the best storage option for 11 Nintendo Switch game cards. Hippo forms an L_2 experience titled "Search broadly, then verify every qualifying product." On `t161`, which asks for storage for 23 cards, Hippo retrieves the experience from `t158`. On this target task, Hippo succeeds in 8/8 attempts, compared with 0/8 for No Memory. This case illustrates reuse of experience when the capacity requirement changes within the same task template.

Relation to the overall method. Together, these cases illustrate complementary functions of the experience hierarchy. L1 preserves the diagnoses associated with individual attempts, including different local recommendations, while L2 provides a retrievable procedure with an explicit scope. The outcomes show where those procedures are reused successfully and where their assumptions or execution remain incomplete. These are qualitative examples with paired task outcomes, rather than an isolated comparison of L2 retrieval against direct L1 retrieval. They explain the role of the design without assigning the entire observed improvement to any single clause or component.

G Recorded Prompt Excerpts

The excerpts below document the inputs and outputs for the cases in Sections D.6 and F.3. Bracketed ellipses mark editorial omissions; plain ellipses preserve omissions in the supplied records. Compact traces and interaction histories are not reproduced. The separate no-memory diagnosis for `t143` illustrates the judge's feedback format. It is not the source of the earlier `t141` experience.

Judge input and output `t143`, no-memory rollout 0

Recorded user fields
`TASK`: How much I spent on home decoration shopping during 1/29/2023
`REFERENCE ANSWER (ground truth):`
`{"must_include": ["265.69"]}`
`OFFICIAL GRADER VERDICT: INCORRECT`
`AGENT FINAL ANSWER: $260.69`
Trace (omitted). Per-step thoughts, actions, and page text.
Judge output (verbatim reason; outcome: `failure`). "The agent correctly identified the $260.69 home-decoration item but reported only its merchandise subtotal. It omitted the item's $5 share of the $10 shipping charge across the two ordered items; adding that yields the required $265.69."

G.1 Writers: reflection and experience construction

The L_1 writer uses an individual trace and its diagnosis. The L_2 writer then compares the outcomes and the available reflections or success traces. The following two inputs both come from `t141`.

L1 writer `t141`, memory-arm rollout 0; system excerpts

You are an expert web-navigation analyst distilling a lesson from ONE failed rollout. It FAILED: The agent undercounted the mixed March 11 order. It included only the $11.43 cornbread mix and missed the $7.99 cake-topper item as food-related, while also failing to include the applicable share of that order's $25 shipping. ...
Extract lessons that would prevent this failure on future tasks on this site. [...]
A single attempt cannot tell a bad route from a good route walked badly: do NOT write blanket prohibitions ("never use X", "page Y is unreliable") from one trajectory — state what to VERIFY on that route instead. [...]
NEVER copy task-specific values (product names, search terms, dates, numbers). [...]
- Return at most 1 item — the single most important, reusable lesson; [] only if nothing reusable was learned.

L2 writer `t141`; system and user excerpts

System. An agent made N independent PARALLEL rollouts of the SAME web task. For each rollout you see its verdict (OK/WRONG, with the judge's reason) and EITHER the lesson distilled from it (its L1) OR, for a genuine success we want to learn from, its full TRACE. Look across ALL N outcomes and the success/failure ratio, then write ONE higher-level lesson: [...]
Excerpt note. The all-failure instruction is omitted here. The prompt also includes "NEVER copy task-specific values (product names, search terms, dates, numbers)."
User. `SITE: shopping`
`TASK`: How much I spent on food-related shopping during March 2023
8 rollouts, 0 succeeded:
— rollout 1 [WRONG] (The agent undercounted the mixed March 11 order. [...])
L1 lesson: Audit mixed orders and allocate shipping: From My Account, select "View All" ...
— rollout 2 [WRONG] (The agent undercounted by treating only the cornbread mix as food-related in the mixed order and using its $11.43 merchandise subtotal without shipping ... It also needed to include the $7.99 cake topper ...)
L1 lesson: Use full order details and include attributable costs: ...
— rollout 3 [WRONG] (... omitted the $5-per-item shipping attributable to each of the three relevant products ...)
L1 lesson: Audit full order details and attributable costs: ...
Excerpt note. The other five rollout entries are omitted.

G.2 Selector and actor: experience use on a later task

The selector receives the current task and candidate experiences. It can choose one or return -1 to select none. The actor receives the selected experience as an optional hint.

Applicability gate `t143`; system and candidate excerpts

System. Given the CURRENT web task and a numbered list of candidate past lessons, pick the ONE most APPLICABLE lesson. It must genuinely fit this task's goal and the controls/pages it needs — matching topic words is NOT enough [...] If NONE genuinely applies, return -1 — injecting nothing is better than a misleading lesson.
Respond JSON {`"idx": int`} (the 0-based index, or -1).
Candidate 0. [0] Audit complete orders and allocate related shipping: Open "My Account" → "My Orders," review all pages ...
Output. The other four candidates are omitted. The gate returns {`"idx": 0`}; cosine similarity is 0.585.

Actor `t143`; shared optional-hint excerpts

Optional hints from past tasks on this site — use one ONLY if it clearly fits the current task, and prefer the cheapest sufficient evidence. Do not add steps or chase a procedure a hint describes if you can already answer directly and correctly:
[strategy] Audit complete orders and allocate related shipping: Open "My Account" → "My Orders," review all pages for the requested period, and select "View Order" on every qualifying row. [...] then add each item's attributable share of "Shipping & Handling" based on the visible per-item or per-unit pattern and reconcile the result with the order totals. [...]
Use. All eight rollouts receive this optional experience. The interaction history is omitted.

G.3 Memory update policies

We compare three policies for incorporating new experience after cross-rollout comparison.

Append only. Each new experience is added to memory. Existing entries remain unchanged, without removal or integration.

Repeated rewrite. The model receives existing memories and a new observation. For each old entry, it can retain the text, revise it to add detail or correct an error, merge it with semantically duplicate entries, or delete it. It can also add facts that existing entries do not cover. An entry can remain unchanged; the policy does not require every entry to be rewritten at every update. The update instruction is:

> Given the existing memories and a new observation, decide for each old entry whether to **keep** it, **rewrite** it (fill in detail or correct errors), **merge** it with semantically duplicate entries into a single item, or **delete** it (falsified, stale, or fully redundant), and add any facts from the new observation that no existing entry covers.

Hippo. Hippo compares new experience with related memories. It discards duplicate candidates and resolves overlaps or conflicts through explicit conditions of use. When new guidance replaces an existing entry, Hippo adds the replacement and deletes the old entry. Entries outside the deletion set retain their original text. This policy allows prior guidance to change while preserving entries that the update does not replace.

H Limitations

Hippo incurs additional inference and interaction costs from multiple rollouts and memory updates. KV-cache reuse in SGLang (Zheng et al., 2024) and vLLM (vLLM Project, n.d.) can avoid recomputing identical prompt prefixes across rollouts, reducing redundant inference computation. Specialized models such as Jev (Almeida, 2026) could handle judgment labels, while a generative model would still produce textual diagnoses. We have not evaluated Jev or quantified the end-to-end savings from KV-cache reuse.

Hippo uses the same base model for task execution, trajectory assessment, and memory updates. Shared blind spots may allow incorrect or overly general lessons to persist in memory. A stronger model could construct and revise experience for a smaller model to apply during execution. We

have not evaluated whether such experience transfers reliably or whether its reuse offsets the added construction cost.

Our long-horizon evaluation studies memory updates across tasks; it does not establish performance on exceptionally long individual tasks. Longer trajectories may strain context budgets and make it harder to retain early decisions and dependencies across stages. Repeating complete rollouts would also increase the cost of collecting experience. Reflection at intermediate stages and hierarchical consolidation are possible extensions, but whether they preserve enough evidence to derive reliable experience remains untested.