MEM-EVOLVER: MEMORY-LEVEL POLICY IMPROVEMENT FOR LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Experience memory enables LLM agents to reuse past interactions without updating model parameters, but growing memory banks introduce a selection bottleneck: semantically similar memories may fail or cause negative transfer when the current phase, object, tool, destination, or precondition differs. We propose MEM-EVOLVER, a no-gradient framework that improves the external memory-selection policy rather than simply expanding the memory bank. MEM-EVOLVER builds compact experience cards from success–failure divergences, records historical outcomes of memory use as structured usage records, and reranks retrieved candidates using evidence of utility, negative transfer, and transfer boundaries. This converts environmental feedback into reusable evidence about which memories should be trusted under compatible decision states. Experiments on ALFWorld and ScienceWorld show improvements over no-memory agents, representative memory baselines, and semantic retrieval across multiple backbones. Ablations verify the roles of usage records and boundary evidence, while record-accumulation and retrieval-quality analyses show that usage-aware reranking recovers useful memories missed by semantic top-1 retrieval. Our code is available at https://anonymous.4open.science/r/MemEvolver-81F0.

1 INTRODUCTION

LLM agents can reason, use tools, and interact with external environments, yet their useful interaction histories are often left as one-off traces rather than reusable capability (Wei et al., 2022; Yao et al., 2023; Schick et al., 2023; Qin et al., 2023; Park et al., 2023). Updating model parameters can improve behavior through supervised fine-tuning, reinforcement learning, preference optimization, or self-play (Ouyang et al., 2022; Bai et al., 2022; Rafailov et al., 2023; Yuan et al., 2024; Chen et al., 2024), but such pipelines add deployment cost and can be difficult to run continuously for interactive agents. A complementary direction is to keep the backbone fixed and improve the external context it receives. Experience memory stores past successes, failures, and recovery patterns outside the model for reuse in future episodes (Zhang et al., 2025; Li et al., 2025).

Prior memory-based agents demonstrate that externalized experience can improve behavior. ExpeL distills lessons from trials (Zhao et al., 2024); Voyager stores discovered skills (Wang et al.); ReasoningBank scales reasoning memory (Ouyang et al., 2026); FLEX studies continuous agent evolution from experience (Cai et al., 2025); Memento retrieves prior cases (Zhou et al., 2025); and long-term, structured, reflection, and workflow memories organize persistent interaction information in different forms (Packer et al., 2023; Zhong et al., 2024; Shinn et al., 2023; Xu et al., 2025; Zeng et al., 2024; Wang et al., 2025). These methods mainly study what to write into memory, how to compress it, or how to retrieve seemingly relevant entries. As the memory bank grows, however, a different bottleneck becomes central: *the agent must decide which retrieved experience should be trusted for the current decision state*. A memory may mention similar goals or objects but still be harmful when its phase, target object, destination, tool, precondition, quantity constraint, or executable action set does not match the present state. Thus, relevance is not the same as utility.

Semantic retrieval is useful for narrowing the candidate set, but it cannot by itself determine whether applying a memory will improve behavior. This motivates a retrieve-then-rerank view (Sun et al., 2023; Qin et al., 2024) in which semantic retrieval identifies relevant candidates and reranking assesses their *contextual utility* by judging how useful each experience is likely to be in the current

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.

usage context. Because the same experience can help in one context but fail in another, this judgment draws on outcomes and transfer boundaries recorded in previous uses under compatible conditions (Sec. 2.3). Figure 1 illustrates the issue on ALFWorld: as the experience bank grows, Recall@10 continues to expose substantially more recoverable successes than Recall@1, leaving a large gap between candidate availability and the memory actually selected by top-1 retrieval. The agent therefore needs evidence of memory utility and negative transfer under compatible conditions.

We propose MEM-EVOLVER, a no-gradient framework for memory-strategy improvement in LLM agents. The core idea is to improve the external experience-selection policy rather than simply expanding the memory bank. MEM-EVOLVER contains three components. First, it constructs compact experience cards from success–failure divergences, focusing each memory on the local behavioral difference that changed the outcome. Second, it performs record-aware memory selection: semantic recall proposes candidate experiences, while historical usage records provide evidence for list-wise reranking. Third, it converts post-rollout feedback into structured usage records containing the usage context, feedback label, reward difference, and mismatch or transfer-boundary annotations. Usage records guide selection with a fixed backbone.

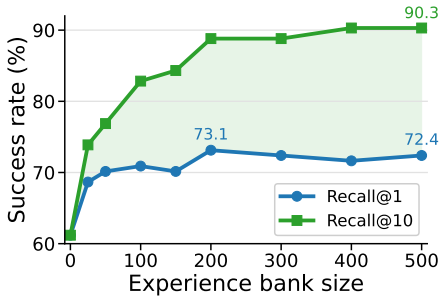


Figure 1: As the experience bank grows, a persistent gap remains between retrieval Recall@1 and Recall@10.

We evaluate MEM-EVOLVER on ALFWorld (Shridhar et al., 2021) and ScienceWorld (Wang et al., 2022b). The main results show that MEM-EVOLVER improves over no-memory agents, representative memory baselines, and semantic retrieval across multiple backbones. Using the success-rate cells reported in the main table, MEM-EVOLVER yields a 46.51% relative improvement over the no-thinking baseline and a 5.41-point gain over the strongest prior memory baseline. Ablations confirm that usage records, boundary evidence, and robust selection each contribute to the final selector. Record-accumulation analysis shows that performance improves as more historical usage feedback becomes available, while retrieval-quality analysis shows that retrieved candidates contain useful memories that semantic top-1 selection fails to exploit.

Our contributions are threefold:

- We formulate experience use as a memory-level selection problem and propose MEM-EVOLVER, a no-gradient framework that improves which external memory an LLM agent trusts.
- We introduce usage-record-guided selection that combines experience construction, record-aware reranking, and feedback updates to assess memory utility and transfer boundaries.
- We validate the method on ALFWorld and ScienceWorld with multiple backbones, showing improvements over no-memory agents, memory baselines, and semantic retrieval, together with ablations and diagnostics that explain where the gains come from.

2 MEM-EVOLVER FRAMEWORK

Figure 2 illustrates the overall framework of MEM-EVOLVER. The method improves experience-memory use through three connected modules. Divergence-Guided Memory Construction extracts reusable experience cards from success–failure divergences in the agent's own rollouts. Record-Aware Memory Selection retrieves candidate experiences and reranks them with historical usage evidence. Feedback-Driven Evidence Update converts rollout feedback into usage records that guide future memory selection. The model parameter remains fixed throughout the process, while adaptation is achieved through external memory selection strategies.

2.1 PROBLEM FORMULATION

Memory-based LLM agents typically retrieve a past experience and insert it into the reasoning prompt (Shinn et al., 2023; Zhao et al., 2024; Zhang et al., 2023). In interactive environments, semantic relevance does not necessarily imply behavioral utility. A memory that appears similar to the

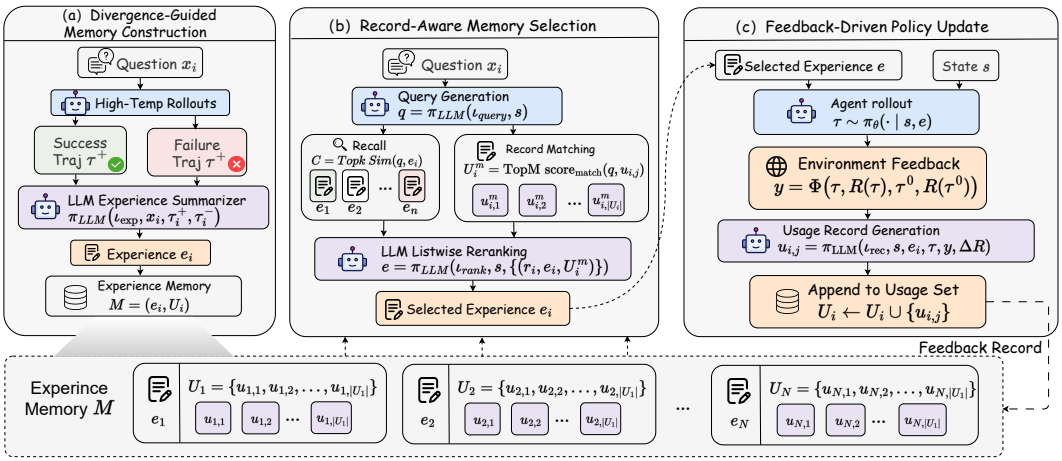


Figure 2: Overview of MEM-EVOLVER. The query and semantic candidate set in (b) are computed once at episode initialization from the task and initial observation. (a) Divergence-Guided Memory Construction builds reusable experience cards from successful and failed rollouts. (b) Record-Aware Memory Selection retrieves candidate experiences and reranks them with matched usage records. (c) Feedback-Driven Evidence Update writes rollout feedback into usage records, which guide future memory selection without changing the backbone model.

current task may still be harmful when it mismatches the current phase, object, destination, precondition, tool, count, or executable action set. MEM-EVOLVER treats experience use as a memory-level decision problem with an adaptive external selection policy.

Let π_θ denote the fixed reasoning policy. At decision step t, the agent observes a decision state $s_t = (x, o_t, h_t, \mathcal{A}_t)$, where x is the task goal, o_t is the current observation, h_t is the recent action-observation history, and $\mathcal{A}_t$ is the admissible action set when available. The external memory system is $(\mathcal{M}, \mathcal{U})$. The experience bank is $\mathcal{M} = \{e_i\}_{i=1}^N$, where each e_i is a reusable experience card. The usage-record bank is $\mathcal{U} = \{\mathcal{U}_i\}_{i=1}^N$, where $\mathcal{U}_i$ stores historical feedback records associated with e_i. This separation distinguishes *what* can be reused from *when* that reuse has helped, failed, or transferred poorly. Appendix B.1 gives a concrete card–record example that distinguishes episode context from first-citation evidence. This view is related to semi-parametric agent formulations, where external memory changes the policy context without directly changing model parameters (Zhang et al., 2023).

Given s_t, the memory-selection policy $\mu_{\mathcal{U}}(e \mid s_t; \mathcal{M})$ selects an experience from the bank. After selecting e_t, the reasoning policy executes a memory-conditioned continuation rollout $\tau_{t:T}^e \sim \pi_\theta(\cdot \mid s_t, e_t)$ and receives return $R(\tau_{t:T}^e)$. The ideal utility-oriented memory-selection policy is

$$\mu_{\mathcal{U}}^\star = \arg\max_{\mu} \mathbb{E}_{s_t \sim \mathcal{D},\, e_t \sim \mu(\cdot \mid s_t; \mathcal{M}, \mathcal{U}),\, \tau_{t:T}^e \sim \pi_\theta(\cdot \mid s_t, e_t)} \left[R(\tau_{t:T}^e) \right]. \tag{1}$$

MEM-EVOLVER does not optimize Eq. 1 through gradient updates. It implements $\mu_{\mathcal{U}}$ as a non-parametric retrieve-then-rerank policy. As usage records accumulate, the selector receives richer evidence, and future memory decisions can improve without changing π_θ.

The following modules instantiate this objective by constructing reusable experience cards, selecting candidate memories with usage evidence, and updating the evidence bank from rollout feedback.

2.2 DIVERGENCE-GUIDED MEMORY CONSTRUCTION

Memory construction aims to transform raw interaction history into compact guidance that can transfer to future tasks. Full trajectories are often noisy because they contain incidental observations, route choices, and environment-specific details. Successful trajectories alone also provide limited contrast because they do not expose which local decision avoided failure. MEM-EVOLVER therefore constructs experience cards from success–failure divergences produced by the same reasoning policy on the same problem. This module corresponds to Figure 2(a). Following work on learning

from failures and corrections (Shinn et al., 2023; Zhao et al., 2024), success–failure comparisons produce compact experience cards instead of full trajectories.

For each memory-construction problem $x_i \in \mathcal{D}_{\text{mem}}$, the agent samples multiple rollouts under an exploratory decoding setting. When both a successful trajectory τ_i^+ and a failed trajectory τ_i^- are available, the LLM Experience Summarizer prompt ι_{exp} compares them and produces one structured experience card

$$e_i = \pi_{\text{LLM}}\left(\iota_{\text{exp}}, x_i, \tau_i^+, \tau_i^-\right), \qquad \mathcal{U}_i \leftarrow \emptyset. \tag{2}$$

Each experience card stores the task stage, objective, trigger condition, reusable behavioral principle, referable action pattern, failure risk, and transfer boundary. Problems without a successful–failed divergent pair are skipped because no reliable local contrast can be extracted.

This construction makes the memory bank agent-adaptive. Since both trajectories are generated by the same reasoning policy, the extracted experience reflects the agent's own reasoning habits, action preferences, and failure modes. The resulting card provides reusable guidance, while its initially empty record set will later collect evidence about where the guidance is useful.

2.3 Record-Aware Memory Selection

Memory selection aims to choose one experience that is useful for the current decision state. Semantic recall can efficiently narrow the search space, but it cannot determine whether a retrieved experience has previously improved behavior under compatible conditions [cite]. Mem-Evolver combines semantic recall, record matching, and listwise reranking (Figure 2(b)).

Episode-level semantic recall. At the beginning of each episode, the Query Generation prompt ι_{qry} summarizes the task goal x and the public initial observation o_0 into one structured query:

$$q = (g, c, b) = \pi_{\text{LLM}}(\iota_{\text{qry}}, x, o_0). \tag{3}$$

The three fields are **Goal** (g): the requested operation and target outcome; **Current State** (c): task-relevant facts available at episode initialization; and **Constraint or Bottleneck** (b): explicit requirements or information gaps supported by the input. Here, "current" refers to the initial state, not a continuously updated step-level summary. Appendix E details the fields and stored example.

The fields are serialized in the fixed order (g, c, b) and encoded together by E5, without field-specific weights. The recall stage retrieves the top-k semantically similar experience cards:

$$C = \text{TopK}^k_{i\in\{1,\dots,N\}} \,\text{sim}\left(\text{Emb}(q), \text{Emb}(e_i)\right), \tag{4}$$

where $\text{sim}(\cdot,\cdot)$ is cosine similarity and $\text{Emb}(e_i)$ denotes the retrieval index associated with card e_i. The query and candidate set are computed once and reused throughout the episode; later observations, actions, and outcomes do not regenerate q or trigger full-bank retrieval.

Record matching. Each recalled candidate has historical usage records associated with their source episodes. For each $i \in C$, the selector retrieves the top-m records from $\mathcal{U}_i$:

$$\mathcal{U}^m_{i,t} = \text{TopM}^m_{u\in\mathcal{U}_i} \,\text{score}_{\text{match}}(q, u). \tag{5}$$

If $\mathcal{U}_i$ is empty, then $\mathcal{U}^m_{i,t} = \emptyset$. Let q_u be the episode-start triplet stored with record u. Matching uses the same whole-triplet representation:

$$\text{score}_{\text{match}}(q, u) = \text{sim}\left(\text{Emb}(q), \text{Emb}(q_u)\right). \tag{6}$$

This score measures episode-context similarity; it does not use future outcomes as query features. The local decision state s_t remains a separate input to the downstream selector and reasoning policy.

Listwise reranking. Following LLM-based reranking methods (Sun et al., 2023; Qin et al., 2024; Reddy et al., 2024), we rerank the retrieved candidates by their *contextual utility*, using outcomes and transfer boundaries in matched usage records to assess how useful each experience is likely to be in the current decision context. Episode-context matching identifies relevant historical evidence; the reranker interprets that evidence together with the current decision state to judge which experience to trust. It jointly compares candidate experiences, original recall ranks, and matched records

$$i_t^\star = \pi_{\text{LLM}}\left(\iota_{\text{rank}}, s_t, \left\{\left(r_i, e_i, \mathcal{U}^m_{i,t}\right)\right\}_{i\in C}\right), \qquad e_t = e_{i_t^\star}, \tag{7}$$

Algorithm 1 MEM-EVOLVER Record-Aware Memory-Level Improvement

Require: Frozen reasoning policy π_θ, prompted LLM policy π_{LLM}, experience bank $\mathcal{M} = \{e_i\}_{i=1}^{N}$, usage-record bank $\mathcal{U} = \{\mathcal{U}_i\}_{i=1}^{N}$, task goal x, initial observation o_0, decision state s_t, candidate count k, record count m, return function R, feedback function Φ, update flag $\eta_{\mathrm{upd}} \in \{0, 1\}$.
Ensure: Selected experience e_t, environment action a_t, updated usage-record bank $\mathcal{U}'$.

Episode initialization (once):
$q \leftarrow \pi_{\mathrm{LLM}}(\iota_{\mathrm{qry}}, x, o_0)$ ▷ Eq. 3
$C \leftarrow$ top-k candidate indices from $\mathcal{M}$ by Eq. 4
At decision step t (reuse q and C):
for each candidate index $i \in C$ **do**
  $\mathcal{U}_{i,t}^{m} \leftarrow$ top-m matched records from $\mathcal{U}_i$ by Eq. 5
end for
$i_t^\star \leftarrow \pi_{\mathrm{LLM}}\big(\iota_{\mathrm{rank}}, s_t, \{(r_i, e_i, \mathcal{U}_{i,t}^{m})\}_{i \in C}\big)$ ▷ Eq. 7
$e_t \leftarrow e_{i_t^\star}$
Build a REACT-style prompt with s_t and e_t
$a_t \sim \pi_\theta(\cdot \mid s_t, e_t)$; execute a_t and obtain rollout $\tau_{t:T}^{e}$
$\mathcal{U}' \leftarrow \mathcal{U}$
if $\eta_{\mathrm{upd}} = 1$ and feedback is available **then**
  Obtain $\tau_{t:T}^{0}$ if required; compute y_t and ΔR_t by Eq. 8–9
  $u_{i_t^\star,t} \leftarrow \pi_{\mathrm{LLM}}(\iota_{\mathrm{rec}}, s_t, e_t, \tau_{t:T}^{e}, y_t, \Delta R_t)$ ▷ Eq. 10
  $\mathcal{U}'_{i_t^\star} \leftarrow \mathcal{U}'_{i_t^\star} \cup \{u_{i_t^\star,t}\}$ ▷ Eq. 11
end if
return $e_t, a_t, \mathcal{U}'$

where r_i is the original semantic recall rank. The reranker promotes candidates with positive historical utility under compatible phase, object, destination, tool, count, state transition, and required-chain conditions. It demotes candidates with negative transfer or hard boundary mismatches. When record evidence is sparse or ambiguous, the original retrieval rank serves as a conservative prior.

During selection, the reranker uses only records available before the current decision. It does not access the current rollout, future observations, rewards, or success labels. The selected experience is inserted into the ReAct-style reasoning prompt as optional local guidance.

2.4 FEEDBACK-DRIVEN EVIDENCE UPDATE

Usage feedback provides evidence about experience utility beyond the stored cards themselves (Zhang et al., 2026). Aggregating feedback from all usage records of an experience card can obscure which records are relevant to the current problem. Its contextual utility is therefore estimated from the most relevant usage records. Each new record therefore retains both the feedback and its usage context. This update (Figure 2(c)) follows feedback-driven agent improvement (Shinn et al., 2023; Madaan et al., 2023; Gou et al., 2023) and fine-grained feedback for memory management (Ma et al., 2026).

After the selector chooses $e_t = e_{i_t^\star}$, the reasoning policy executes a memory-conditioned continuation rollout

$$\tau_{t:T}^{e} \sim \pi_\theta(\cdot \mid s_t, e_t), \qquad y_t = \Phi\left(\tau_{t:T}^{e}, R(\tau_{t:T}^{e}), \tau_{t:T}^{0}, R(\tau_{t:T}^{0})\right). \tag{8}$$

Here $\tau_{t:T}^{0}$ denotes a memoryless or baseline continuation rollout when such comparison is used. The feedback label y_t summarizes positive effects, failure or negative transfer, and no measurable effect. The return difference is

$$\Delta R_t = R(\tau_{t:T}^{e}) - R(\tau_{t:T}^{0}). \tag{9}$$

The Usage Record Generation prompt ι_{rec} combines this feedback with the usage context to form a structured record

$$u_{i_t^\star,t} = \pi_{\mathrm{LLM}}\left(\iota_{\mathrm{rec}}, s_t, e_t, \tau_{t:T}^{e}, y_t, \Delta R_t\right) = \left(\phi_s(s_t), \phi_e(e_t), \psi(\tau_{t:T}^{e}), y_t, \Delta R_t, b_t\right), \tag{10}$$

where $\phi_s(s_t)$ stores structured state fields, $\phi_e(e_t)$ stores fields of the selected experience, $\psi(\tau_{t:T}^{e})$ summarizes realized behavior, and b_t stores mismatch or transfer-boundary annotations. The record

Table 1: Overall performance of MEM-EVOLVER and baselines under the same memory interface, measured by success rate (%). Primary paired comparisons use the significance threshold $p < 0.05$. The Average column is computed over all reported success-rate cells. Rel. Impv. is the relative improvement over the No-thinking average.

Type	Method	ALFWorld			ScienceWorld			Avg.	Rel. Impv.
		Qwen3-8B	GPT-4o-mini	GPT-4o	Qwen3-8B	GPT-4o-mini	GPT-4o		
No memory	No-thinking	60.50	43.30	63.43	26.50	21.80	34.62	41.69	0.00
	REACT	67.91	50.00	64.18	35.38	32.31	36.92	47.78	+14.61
Memory	Memento	69.40	57.46	65.67	41.54	40.00	44.62	53.12	+27.40
	ExpeL	70.90	52.99	63.43	40.00	41.54	48.46	52.89	+26.85
	MemoryBank	72.39	54.48	67.91	43.08	44.62	51.54	55.67	+33.53
	MEM-EVOLVER	**79.85**	**59.70**	**74.63**	**52.31**	**46.15**	**53.85**	**61.08**	**+46.51**

is attached to the selected experience

$$\mathcal{U}_{i_t^\star} \leftarrow \mathcal{U}_{i_t^\star} \cup \{u_{i_t^\star,t}\}. \tag{11}$$

The appended record supports subsequent contextual-utility assessment in Eq. 7 while leaving the reasoning policy π_θ and experience card e_t unchanged.

Algorithm 1 summarizes memory selection and record updates.

3 EXPERIMENTS

3.1 EXPERIMENTAL SETUP

Datasets and Metrics. We evaluate MEM-EVOLVER on two interactive text environments: ALFWorld (Shridhar et al., 2021), which instantiates household instruction-following tasks with admissible actions, and ScienceWorld (Wang et al., 2022b), which evaluates scientific problem solving through text-based environment interaction. ALFWorld is evaluated by task success rate (SR). ScienceWorld is evaluated by success rate. The average score in Table 1 is computed over all reported success-rate cells under the same memory interface. For selector diagnostics, we additionally report Rerank@1 on a failure-correction analysis set, where the base REACT agent fails and memory selection is analyzed as a correction mechanism.

Implementation Details. We construct the experience bank and usage-record bank from the training split of each environment. Qwen3-8B (Yang et al., 2025) is used to generate training rollouts, summarize successful/failed trajectory divergences into structured experience cards, and write usage records from historical feedback. Unless otherwise stated, main evaluation uses the training-split experience bank and usage records only; no test-task success labels or rewards are written back during evaluation. Semantic recall uses the E5 embedding model (Wang et al., 2022a) with cosine similarity to retrieve top-10 candidate experiences; the reranker uses the top-3 matched usage records for each candidate. For each inference row, the same backbone is used for query generation, memory selection, and ReAct-style environment interaction. The primary backbone is Qwen3-8B, and we additionally evaluate GPT-4o and GPT-4o-mini (Hurst et al., 2024) to test cross-backbone transfer of Qwen3-8B-constructed memories. The full MEM-EVOLVER selector combines historical usage records, boundary evidence, robust frozen judgments, parse-valid gating, and the original retrieval rank as a stabilizing prior. At inference time, memory is used as a corrective reference when ReAct-style reasoning shows failure signals, rather than being forced into every step. The No-thinking baseline generates the next action without external memory or ReAct-style reasoning.

3.2 OVERALL PERFORMANCE

To verify that memory-selection improvement translates into real environment performance, we compare MEM-EVOLVER with no-memory baselines and prior memory baselines in live rollouts. Table 1 reports the resulting live-rollout performance across environments, backbones, and memory-use settings. No-memory baselines test the backbone without external experience. Memory baselines test whether storing lessons, cases, or long-term memories is sufficient when the memory

Table 2: Usage-record ablations show which evidence sources support memory selection. FC-Rerank@1 is measured on the failure-correction analysis set, where the base REACT agent fails. Overall SR is measured on the full evaluation set. ΔSR reports percentage-point improvement in Overall SR over the semantic Recall@1 selector.

Variant	ALFWorld			ScienceWorld			Avg. ΔSR
	FC-Rerank@1	Overall SR	ΔSR	FC-Rerank@1	Overall SR	ΔSR	
Recall@1 / semantic selector	28.85	72.39	0.00	11.90	43.08	0.00	0.00
+ LLM judge, no records	34.62	74.63	+2.24	19.05	47.69	+4.61	+3.43
+ Usage records, single judge	36.54	75.37	+2.98	26.19	52.31	+9.23	+6.11
+ Positive-only records	30.77	73.13	+0.74	21.43	49.23	+6.15	+3.45
MEM-EVOLVER	48.08	79.85	+7.46	26.19	52.31	+9.23	+8.35

interface still relies on semantic retrieval to choose the reference. MEM-EVOLVER uses the same broad retrieve-and-inject interface, but replaces relevance-only selection with usage-record guided memory selection. This comparison separates backbone, memory, and selection effects.

Overall gains over memory baselines. MEM-EVOLVER obtains the strongest live-rollout performance among the compared methods in Table 1. On the averaged SR cells, MEM-EVOLVER reaches 61.08, compared with 55.67 for the strongest prior memory baseline and 47.78 for REACT. This corresponds to a 5.41-point gain over the strongest memory baseline and a 13.30-point gain over the no-memory REACT agent. Because the average is computed over all reported success-rate cells under the same interface, the comparison reflects the full cross-environment and cross-backbone setting. The result shows that the improvement is not explained by simply adding an experience bank or switching to a different memory format. It supports the main claim that experience memory needs a stronger selection strategy, not only better storage.

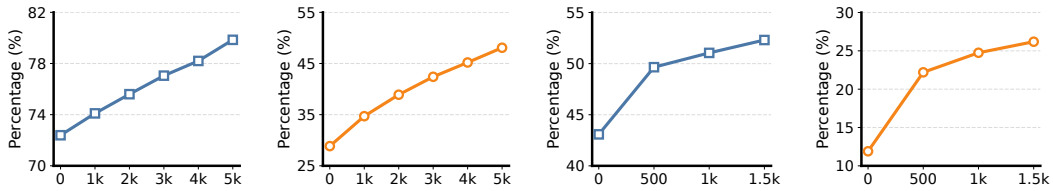
Gains beyond semantic memory retrieval. Memento, ExpeL, and MemoryBank instantiate different memory formats, but their deployed retrieval step is still primarily semantic: a past lesson, case, or memory is selected because it appears relevant to the current state. MEM-EVOLVER improves over these semantic-retrieval memory mechanisms on the averaged SR comparison. This indicates that the key benefit is not only the content stored in memory, but the ability to decide whether a retrieved experience has historically transferred well under compatible local conditions.

Transfer across backbones. The same Qwen3-8B-constructed experience bank and usage records also improve stronger backbones. On GPT-4o-mini, MEM-EVOLVER improves over REACT and outperforms representative memory baselines across the reported rows. On GPT-4o, it remains above the strongest memory baseline. This suggests that usage records capture experience utility beyond the model used to construct the bank. Because the memory resources are shared, the gains on stronger backbones do not require regenerating experience cards with those models.

3.3 ABLATION STUDY

To identify why usage records improve selection, Table 2 ablates the selector from semantic retrieval toward the full MEM-EVOLVER module. The experiment removes or simplifies one source of evidence at a time: historical records, negative and boundary evidence, robust judging, parse-valid filtering, and the retrieval-rank prior. This evaluates RQ2 by testing whether the final performance comes from usage evidence itself or from generic LLM reranking over candidate cards.

The ablation progression in Table 2 shows that card-only judging cannot replace historical feedback. A no-record LLM judge recovers part of the gap over Recall@1, which means the candidate experience cards contain useful signals. Adding real usage records closes more of the gap, showing that prior outcomes provide stronger evidence than experience text alone. The complete selector remains the most robust setting across environments, indicating that usage records are most effective when combined with boundary checks, robust judgments, parse-valid filtering, and the original retrieval rank. Positive history alone is also insufficient. In Table 2, the positive-only variant removes most of the gain on ALFWorld and remains below the full selector on ScienceWorld. This supports a



(a) ALF SR (b) ALF Rerank@1 (c) SW SR (d) SW Rerank@1

Figure 3: Accumulated usage records improve a fixed memory selector. The backbone model, experience bank, retriever, reranker prompt, and decoding setting are fixed. Only the usage-record budget changes. Overall SR measures full-task success, and Rerank@1 measures selector success on the failure-correction analysis set.

central design choice of MEM-EVOLVER: usage records should store not only where an experience helped, but also where it failed or transferred poorly. Negative and boundary evidence helps demote memories associated with wrong objects, inappropriate stages, or repeated lack of progress. Such records qualify the apparent usefulness of a semantically relevant card by identifying conditions under which it failed. Their role in contextual utility is to inform whether those conditions also apply to the current problem.

3.4 THE EFFECT OF USAGE-RECORD SCALE

To test whether MEM-EVOLVER improves through accumulated feedback rather than hidden model changes, we fix the backbone model, experience bank, retriever, reranker prompt, and decoding setting. Only the number of historical usage records available to the selector changes. When a lower-ranked candidate lacks matched records under the current budget, the system falls back to the semantic top-1 experience. This isolates the effect of historical evidence on memory selection.

Figure 3 shows that the no-record setting reduces to semantic top-1 selection, while larger record budgets recover the final MEM-EVOLVER gain. Intermediate budgets allow more lower-rank promotions to become record-backed. Rerank@1 measures selection on the failure-correction set, while overall SR measures complete rollouts, whose later actions may still fail. The gains arise with a fixed backbone and experience bank. Additional records are useful when they provide evidence under comparable conditions; the total record count alone does not describe this coverage.

3.5 RETRIEVAL-ORDER ANALYSIS

Table 3 compares semantic top-1 selection, top-10 coverage, and usage-record-guided reranking on the same failure-correction set. Coverage measures whether useful memories are retrieved; Rerank@1 measures whether the selector identifies them. On ALFWorld, coverage reaches 75.00%, compared with Recall@1 at 28.85%; reranking closes 41.65% of this gap. On ScienceWorld, it closes 50.00% of the gap. These results show that useful memories can be retrieved yet overlooked by semantic ranking.

Table 3: Candidate availability and selected-memory performance.

Env.	R@1	R@10	RR@1	Impv.
ALF	28.85	75.00	48.08	41.65
SW	11.90	40.48	26.19	50.00

Figure 4 decomposes the Rerank@1 gain. ALFWorld yields 21.15% recoveries and 1.92% regressions, a net gain of 19.23 points. ScienceWorld yields 19.05% recoveries and 4.76% regressions, a net gain of 14.29 points. Recoveries exceed regressions in both environments, indicating that gains from promoting useful candidates outweigh losses from replacing the semantic top-ranked experience.

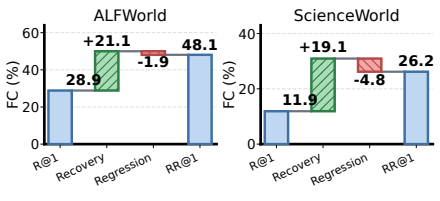


Figure 4: Paired reranking outcomes.

4 Related Work

Self-improving LLM Agents. LLM agents can improve from interaction history through either parameter updates or external adaptation. Parametric approaches update model behavior through instruction tuning (Ouyang et al., 2022), reinforcement learning from feedback (Bai et al., 2022), preference optimization (Rafailov et al., 2023), self-rewarding (Yuan et al., 2024), or self-play (Chen et al., 2024). No-gradient methods instead keep the backbone fixed and adapt through feedback or memory: Reflexion (Shinn et al., 2023) stores verbal reflections, CRITIC (Gou et al., 2023) uses tool-interactive critique, ExpeL (Zhao et al., 2024) distills experience lessons, Voyager (Wang et al.) stores reusable skills, and Agent Workflow Memory (Wang et al., 2025) reuses workflows. These methods show that historical interaction can improve future behavior without updating model parameters. Mem-Evolver follows this lightweight direction, but uses feedback for a different purpose: it improves the external policy that selects which stored experience should be trusted, rather than only revising the current output, adding a reflection, or reusing a workflow.

Experience Memory Selection. Agent memory systems store persistent, episodic, structured, or workflow information, including MemGPT (Packer et al., 2023), MemoryBank (Zhong et al., 2024), Generative Agents (Park et al., 2023), A-MEM (Xu et al., 2025), structural memory (Zeng et al., 2024), and workflow memory (Wang et al., 2025). Experience-memory methods are closest to our setting: ExpeL (Zhao et al., 2024) stores lessons, Voyager (Wang et al.) stores skills, ReasoningBank (Ouyang et al., 2026) stores reasoning memory, FLEX (Cai et al., 2025) learns from experience, and Memento (Zhou et al., 2025) retrieves prior cases. Most of these methods focus on the memory unit itself, including how experiences are written, summarized, structured, or retrieved by semantic similarity. However, interactive environments require deciding whether a memory is useful under the current phase, object, destination, precondition, tool, and action set. LLM reranking methods such as RankGPT (Sun et al., 2023) and PRP (Qin et al., 2024) improve relevance ranking, but experience selection additionally requires evidence about realized utility. Recent work shows that agents may over-follow retrieved experience (Xiong et al., 2025b), while fine-grained feedback can align long-horizon memory use (Ma et al., 2026). ExpeL (Zhao et al., 2024) distills reusable insights from past trials, but retrieves demonstrations by semantic similarity without estimating their reuse utility. MemRL (Zhang et al., 2026) incorporates environmental feedback into utility-aware retrieval, but its per-entry Q-values aggregate past outcomes without reassessing utility from usage records matched to the current context. These approaches leave open how to assess an experience's utility in the current context from the outcomes of its use under comparable conditions.

5 Conclusion

We introduced Mem-Evolver, a no-gradient framework for improving experiential memory use in LLM agents. It addresses a core bottleneck of memory-based agents: as stored experiences grow, the agent must decide which experience should be trusted rather than simply retrieving the most similar one. Mem-Evolver improves this memory-selection strategy by combining three designs. It constructs experience cards from success–failure divergences, records previous experience use as structured usage records, and reranks retrieved candidates with evidence about utility, negative transfer, and transfer boundaries. This converts environmental feedback into reusable selection evidence without updating the backbone model. Contextual utility depends on how closely historical usage conditions match the current problem. Retaining outcomes together with their usage context allows the selector to distinguish relevant evidence from records of the same experience used under different conditions. A successful use under one set of conditions need not support reuse under another, and a failed use can reveal a boundary rather than invalidate the card altogether. Keeping these records separate preserves the evidence needed to make that distinction for later tasks. Experiments on ALFWorld and ScienceWorld show that Mem-Evolver improves over no-memory agents, representative memory baselines, and semantic retrieval. Ablations verify that usage records and boundary-aware evidence are essential to the selector, while record-accumulation and retrieval-quality analyses show that historical feedback helps recover useful experiences missed by semantic top-1 retrieval. These results suggest a practical direction for more reliable memory-based agents: learning from past memory use, not merely storing more memories.

6 AI USE STATEMENT

Large language models are a component of MEM-EVOLVER. During memory construction, an LLM compares successful and failed trajectories to generate experience cards. During memory selection, LLMs generate episode-level queries and rerank candidate experiences using historical usage records. An LLM also converts rollout feedback into structured usage records. The experiments use Qwen3-8B, GPT-4o, and GPT-4o-mini, as described in Section 3.1. AI tools also assist in manuscript preparation. We use ChatGPT and Codex to polish the text, improve clarity and readability, and adjust LaTeX formatting.

REFERENCES

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Zhicheng Cai, Xinyuan Guo, Yu Pei, Jiangtao Feng, Jinsong Su, Jiangjie Chen, Ya-Qin Zhang, Wei-Ying Ma, Mingxuan Wang, and Hao Zhou. Flex: Continuous agent evolution via forward learning from experience. *arXiv preprint arXiv:2511.06449*, 2025.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*, 2024.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang, Nan Duan, Weizhu Chen, et al. Critic: Large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations*, 2023.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Xiaopeng Li, Pengyue Jia, Derong Xu, Yi Wen, Yingyi Zhang, Wenlin Zhang, Wanyu Wang, Yichao Wang, Zhaocheng Du, Xiangyang Li, et al. A survey of personalization: From rag to agent. *ACM Transactions on Information Systems*, 2025.

Weitao Ma, Xiaocheng Feng, Lei Huang, Xiachong Feng, Zhanyu Ma, Jun Xu, Jiuchong Gao, Jinghua Hao, Renqing He, and Bing Qin. Fine-mem: Fine-grained feedback alignment for long-horizon memory management. *arXiv preprint arXiv:2601.08435*, 2026.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

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

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G Patil, Ion Stoica, and Joseph E Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

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.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *The twelfth international conference on learning representations*, 2023.

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, et al. Large language models are effective text rankers with pairwise ranking prompting. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 1504–1518, 2024.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Revanth Gangi Reddy, JaeHyeok Doo, Yifei Xu, Md Arafat Sultan, Deevya Swain, Avirup Sil, and Heng Ji. First: Faster improved listwise reranking with single token decoding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 8642–8652, 2024.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021.

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, 2023.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022a.

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

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *International Conference on Machine Learning*, pp. 63897–63911. PMLR, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V. Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022.

Weimin Xiong, Yifan Song, Qingxiu Dong, Bingchan Zhao, Feifan Song, Xun Wang, and Sujian Li. Mpo: Boosting llm agents with meta plan optimization, 2025a. URL `https://arxiv.org/abs/2503.02682`.

Zidi Xiong, Yuping Lin, Wenya Xie, Pengfei He, Zirui Liu, Jiliang Tang, Himabindu Lakkaraju, and Zhen Xiang. How memory management impacts llm agents: An empirical study of experience-following behavior. *arXiv preprint arXiv:2505.16067*, 2025b.

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

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*, 2024.

Ruihong Zeng, Jinyuan Fang, Siwei Liu, and Zaiqiao Meng. On the structural memory of llm agents. *arXiv preprint arXiv:2412.15266*, 2024.

Danyang Zhang, Lu Chen, Situo Zhang, Hongshen Xu, Zihan Zhao, and Kai Yu. Large language models are semi-parametric reinforcement learning agents. *Advances in Neural Information Processing Systems*, 36:78227–78239, 2023.

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

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

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.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 19724–19731, 2024.

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

A Datasets and Evaluation Setup

ALFWorld. We evaluate on the ALFWorld `eval_out_of_distribution` split used by the adopted environment setup. Each task provides textual observations, admissible actions, and a binary task-success signal. We report task success rate as the main ALFWorld metric.

ScienceWorld. We evaluate on the accepted ScienceWorld test set used by the adopted environment setup. We report task success rate.

Failure-correction analysis set. To study whether memory selection can repair failures of the base reasoning policy, we additionally define a failure-correction analysis set for each environment. This set contains evaluation tasks where the base ReAct rollout fails. It is used only for post-hoc analysis of memory-selection behavior. All main performance claims are reported on the full evaluation set.

Retrieval-quality metrics. Recall@1 denotes using the original semantic top-1 retrieved experience. Recall@10 Coverage denotes whether at least one useful experience exists in the retrieved top-10 candidate set. Rerank@1 denotes the experience finally selected by Mem-Evolver after usage-aware reranking. Recall@10 Coverage is used only as a diagnostic of candidate availability, while Rerank@1 is the deployable selection result.

Protocol consistency. During selection, the reranker receives only the current decision state, retrieved candidate experiences, original retrieval ranks, and historical usage records available before the decision. It does not access future observations, rollout outcomes, rewards, or success labels for the current decision. Success labels are used only after selection for evaluation.

B Implementation Details

Memory construction. We build environment-specific experience banks and usage-record banks from the training split of ALFWorld and ScienceWorld. Qwen3-8B (Yang et al., 2025) is used for memory construction. For each training problem, we sample multiple rollouts with exploration temperature 1.0 and construct an experience card when both a successful trajectory and a failed trajectory are available. The experience summarizer compares the two trajectories and writes a structured experience card containing task phase, goal information, reusable behavioral principle, failure risk, referable action pattern, and transfer boundary. Historical feedback is converted into usage records that store structured state fields, selected experience fields, trajectory summaries, feedback labels, reward differences, and mismatch annotations.

Algorithm 2 MEM-EVOLVER Divergence-Guided Memory Construction

Require: Memory-construction problems $\mathcal{D}_{\text{mem}}$, frozen reasoning policy π_θ, prompted LLM policy π_{LLM}, rollout budget B, exploration temperature T_{exp}, success predicate $\text{succ}(\cdot)$.
Ensure: Experience bank $\mathcal{M}_0$ and usage-record bank $\mathcal{U}_0$.

$\mathcal{M}_0 \leftarrow \emptyset, \mathcal{U}_0 \leftarrow \emptyset$
for each problem $x_i \in \mathcal{D}_{\text{mem}}$ **do**
  Sample rollouts $\mathcal{T}_i \leftarrow \{\tau_i^{(b)} \sim \pi_\theta(\cdot \mid x_i; T_{\text{exp}})\}_{b=1}^{B}$
  $\mathcal{T}_i^+ \leftarrow \{\tau \in \mathcal{T}_i \mid \text{succ}(\tau) = 1\}$, $\mathcal{T}_i^- \leftarrow \{\tau \in \mathcal{T}_i \mid \text{succ}(\tau) = 0\}$
  if $\mathcal{T}_i^+ \neq \emptyset$ and $\mathcal{T}_i^- \neq \emptyset$ **then**
    Select $\tau_i^+ \in \mathcal{T}_i^+$ and $\tau_i^- \in \mathcal{T}_i^-$
    $e_i \leftarrow \pi_{\text{LLM}}(\iota_{\text{exp}}, x_i, \tau_i^+, \tau_i^-)$ ▷ Eq. 2
    if e_i is parse-valid **then**
      $\mathcal{M}_0 \leftarrow \mathcal{M}_0 \cup \{e_i\}, \quad \mathcal{U}_{0,i} \leftarrow \emptyset$
    end if
  end if
end for
return $\mathcal{M}_0, \mathcal{U}_0$

Retrieval and embeddings. Semantic recall uses E5 text embeddings (Wang et al., 2022a). At the beginning of each episode, a structured query is generated from the task goal and public initial observation to retrieve top-$k = 10$ candidate experiences using cosine similarity. The query and candidate set are reused throughout the episode. For each candidate, the reranker receives the top-$m = 5$ available usage records ranked by cosine similarity between the E5 embeddings of the current and historical episode-start triplets. Each triplet is encoded as a whole, without field-specific weights. The final selector uses candidate experience cards, matched usage records, original retrieval ranks, and the current decision state. The original retrieval rank is kept as a conservative prior when the record evidence is weak or ambiguous.

Backbones. Qwen3-8B (Yang et al., 2025) is used for memory construction and as the primary backbone. GPT-4o and GPT-4o-mini (Hurst et al., 2024) are evaluated only at inference time using the same Qwen3-8B-constructed experience bank and usage records.

Inference and prompting. All reasoning runs build on a ReAct-style interaction loop. For step-level memory methods, the selected experience is inserted as optional local decision-memory guidance for the next action. The backbone model is instructed to verify the current object, destination or receptacle, task phase, state, tool, count, and admissible actions before using the memory. If the selected experience does not match the current local state, the model is instructed to ignore the memory and continue with the base ReAct policy. For a given backbone row, the same model is used

for query construction, memory selection, and environment action generation. The main Qwen3-8B runs use deterministic decoding at test time.

Compute. All local experiments are run on a server with 4 NVIDIA A100 GPUs. Memory construction is performed once per environment, and inference experiments reuse the constructed experience and usage-record banks.

B.1 A Concrete Experience Card and Its Usage Evidence

An experience card describes reusable guidance; a usage record describes what happened when that card was available in a particular historical episode. Table 4 illustrates this distinction with a real ALFWorld cooling-task example from our offline reranking diagnostic. The episode-start context used for matching is distinct from the state before the first explicit citation, which is inspected retrospectively as evidence.

Component	Condensed content
Current task	Cool a mug and place it in a cabinet.
Experience card	When the target is not visible, visit and open cabinets to inspect their contents; continue until the target is located, then proceed to cooling and placement. Opening requires being at the cabinet. A historical object location is not a location guarantee for a new episode.
Historical context	**Goal:** cool a mug and place it in a cabinet. **Current state:** the initial observation does not establish the mug's location. **Constraint:** locate and acquire the mug before cooling and placement.
Successful use	**Before first citation:** target location unknown; acquisition and cooling not yet observed. **Trace:** open cabinets (card cited) $\rightarrow$ empty observations $\rightarrow$ ... $\rightarrow$ acquire mug $\rightarrow$ cool mug $\rightarrow$ place mug in cabinet (these final three actions do not cite the card). **Outcome:** success.
Unsuccessful use	**Task:** cool a mug and place it in a cabinet. **Trace:** visit and open cabinet (card cited) $\rightarrow$ mug observed $\rightarrow$ acquire mug (not cited) $\rightarrow$ ... $\rightarrow$ further cabinet searches; an open action returns "Nothing happens." **Outcome:** the episode ends without success at the step limit.
Evidence scope	Citation is observed during search, while successful cooling and placement occur without citation. The unsuccessful episode also acquires the mug, so its final failure cannot be equated with failure to find the target. These records expose stage-specific behavior; neither episode outcome alone establishes the card's causal contribution.

Table 4: One experience card with two historical usage records. Text is condensed from saved evidence; ... marks omitted actions, not inferred events. Instance identifiers and routine navigation are omitted. The historical context and first-citation state belong to past episodes, not the current test episode. This is a representation example, not a controlled attribution test.

C External Resources and Reference Implementations

Table 5 lists the datasets, baseline references, reference experiment protocols, and model or retrieval tools used in our experiments. The links are included for attribution and reproducibility. Our anonymous code release provides the exact run scripts, prompts, and configuration files used to instantiate these resources in our experiments. We do not redistribute third-party datasets, environment packages, model weights, or proprietary model outputs as external assets. Users should obtain external resources from their original sources and comply with the corresponding licenses or service terms. The released artifacts document our own prompts, scripts, memory schemas, and generated experience or usage-record formats.

D Memory-Interface Diagnostic

Mem-Evolver treats a selected experience as local decision evidence for the current state. It is not intended to be a task-level workflow. This distinction matters because workflow-style references can

Table 5: External resources used for datasets, baselines, reference protocols, and implementation components.

Category	Resource	License / terms	Link
Dataset / environment	ALFWorld	MIT License	ALFWorld GitHub
Dataset / environment	ScienceWorld	Apache-2.0 License	ScienceWorld GitHub
Reasoning baseline	ReAct prompt framework	Paper / project page; no code redistributed	ReAct project page
Memory baseline	ExpeL	Apache-2.0 License	ExpeL GitHub
Memory baseline	MemoryBank	MIT License	MemoryBank GitHub
Memory baseline	Memento	MIT License	Memento GitHub
Backbone model	Qwen3-8B	Apache-2.0 model license	Qwen3-8B Hugging Face
Backbone model	GPT-4o	OpenAI service terms	GPT-4o release page
Backbone model	GPT-4o-mini	OpenAI service terms	GPT-4o-mini release page
Retrieval model	E5 text embeddings	MIT License	E5 GitHub

encourage the backbone model to copy a historical procedure and ignore the current object, phase, or admissible-action constraints. Workflow-style references are closer to task-level planning memories such as MPO-style meta plans (Xiong et al., 2025a).

We therefore study a diagnostic variant that injects the reranked experience as a workflow-style reference at every step. This variant is intentionally mismatched with the granularity of MEM-EVOLVER experiences and is not used as the main method. In a sentinel analysis, this variant achieves 10.00% success even though the retrieved-step rate is 1.0. Failure inspection shows over-following, object drift, and repeated no-progress loops. This indicates that poor behavior is caused by the memory-injection interface rather than by failure to retrieve experiences.

The main experiments therefore use a local decision-memory interface for all step-level memory methods. Workflow-level baselines should be reported separately from step-level memory-selection baselines.

E PROMPT DETAILS

This section lists the prompts used by MEM-EVOLVER and keeps the naming consistent with Fig. 2 and Sec. 3. The framework uses the LLM Experience Summarizer prompt ι_{exp}, the Query Generation prompt ι_{qry}, the LLM Listwise Reranking prompt ι_{rank}, Experience-Guided LLM Reasoning for the reasoning agent, and the Usage Record Generation prompt ι_{rec}. The same prompt names are used for both ALFWorld and ScienceWorld; dataset-specific fields only change the serialized input and a small number of domain constraints.

LLM Experience Summarizer Prompt ι_{exp}. This prompt implements Eq. 2. It compares a successful rollout τ_i^+ and a failed rollout τ_i^- from the same training problem x_i and produces one structured experience e_i.

```
You are constructing one reusable experience from two rollouts of the
↪  same task.

Input:
- TASK x_i
```

```
- SUCCESSFUL_ROLLOUT tau_i^+
- FAILED_ROLLOUT tau_i^-

Goal:
Find the key divergence between the successful and failed rollouts and
↪  write
one reusable experience that can guide a future local decision.

Rules:
- Do not summarize the whole trajectory.
- Focus on the local decision where the successful rollout avoids the
↪  failed
  branch.
- Preserve task stage, objective, reusable behavioral principle, failure
↪  risk,
  and referable action pattern.
- Generalize away accidental route details and instance ids.
- Keep concrete target, destination, tool, or state-change fields only
↪  when
  they define the transfer boundary.

Return JSON only:
{
  "task_stage": "...",
  "objective": "...",
  "trigger_condition": "...",
  "behavioral_principle": "...",
  "referable_action_pattern": ["..."],
  "failure_risk": "...",
  "transfer_boundary": "...",
  "evidence_from_success": "...",
  "evidence_from_failure": "..."
}
```

For ScienceWorld, the same ι_{exp} template is filled with ScienceWorld-specific fields such as protocol stage, state cue, successful action, failed action, and final focus or answer action when applicable. The output still represents a single e_i in the experience bank $\mathcal{M}$.

Episode-Level Query Generation ι_{qry}. The query in Eq. 3 is generated once from the task goal and the public initial observation, before the first environment action. It contains exactly three English string fields, shown below. The JSON key `current_state` denotes the episode's initial observable state; it does not imply per-step query regeneration.

Field	Content and grounding rule
`goal`	Requested operation and target outcome, preserving object type, quantity, destination, and desired condition.
`current_state`	Directly observed initial location, relevant objects, access, or explicitly supplied inventory. A requested condition is not an observed fact.
`constraint_or_bottleneck`	Explicit task requirements, ordering, thresholds, or input-supported information gaps. If no bottleneck is established, retain that uncertainty.

The production query specification instructs the summarizer to index rather than solve the task, preserve explicit requirements, and avoid inventing routes, prerequisites, object locations, or inventory. An object not listed in a partial observation is not assumed absent. Task identifiers, rewards, outcomes, trajectories, and future observations are excluded. The generation inputs are `TASK_GOAL`, `INITIAL_OBSERVATION`, and explicitly supplied initial public state, if available.

Stored ALFWorld example. The following triplet (line-wrapped for display) is reproduced from the saved episode-context retrieval audit for the task of placing a clean mug in a coffeemachine. The example uses only the task goal and public initial observation, without the eventual rollout outcome.

```
{
  "goal": "The requested task goal is to put a clean mug in the
    coffeemachine.",
  "current_state": "Room contains coffeemachine 1, cabinets 1-6,
  ↪  countertops
    1-3, drawers 1-3, fridge 1, microwave 1, shelves 1-3, sinkbasin 1,
    stoveburners 1-4, toaster 1. No mug is listed in the initial
    observation.",
  "constraint_or_bottleneck": "The mug must be clean. No explicit
  ↪  ordering,
    focus, or register actions are stated. No mug is listed in the
    ↪  initial
    observation; its current location is not established."
}
```

Embedding and reuse. The three strings are concatenated in a fixed order:

```
query: goal: <goal>
current_state: <current_state>
constraint_or_bottleneck: <constraint_or_bottleneck>
```

E5-base-v2 encodes this entire text into one normalized 768-dimensional vector, with no separate field weights. Historical episode contexts use the same representation. Both the generated query and its recalled candidate set are reused throughout the episode; changing local observations remain separate from this fixed retrieval query.

LLM Listwise Reranking Prompt ι_{rank}. Reranking uses one LLM Listwise Reranking prompt, corresponding to Eq. 7. It receives the decision state s, the retrieved candidates C, their original recall ranks r_i, and the selected usage records $\mathcal{U}_i^m$ for each candidate.

```
You are doing LISTWISE reranking of retrieved experience candidates.
↪  Compare
the entire top-k list before choosing top1.

Do not use hidden success labels, exp_id memorization, or dataset
↪  priors. Use
only the current case, experience cards, original retrieval ranks, and
↪  selected
usage records shown here.

Non-negotiable policy:
1. Treat original retrieval rank as a strong prior. Do not replace
↪  retrieval
   top1 unless another candidate has clearly stronger transferable
   ↪  usage-record
   evidence.
2. Usage records transfer only when task family, target, destination,
↪  tool,
   count, required state, current phase, and required chain are
   ↪  compatible with
   the current case.
3. Full-chain helped-complete evidence is stronger than critical-partial
   acquisition/search evidence. Do not rank search/acquire-only records
   ↪  above
   full-chain records unless the current bottleneck is explicitly
   acquisition/search.
4. Hard mismatches are veto-level risks: wrong task family, wrong
↪  destination,
   wrong tool, wrong state transformation, wrong count, wrong stage, or
   ↪  harmful
   transfer.
5. For state-change tasks, distinguish acquire-target,
↪  transform-with-tool or
   receptacle, and deliver-target. Prefer records covering the missing
   ↪  stage,
```

```
   not merely the first stage.
6. For inspect-with-tool tasks, distinguish locating/acquiring the
↪  target from
   using the tool/device to inspect it.

Listwise procedure:
A. Identify the current bottleneck and required remaining chain.
B. Partition all candidates into strong_transfer, plausible_transfer,
   weak_or_partial, and reject_or_boundary.
C. Within each partition, use original retrieval rank as tie-breaker.
D. Return a complete ranking. The top3 should be a conservative
↪  shortlist:
   include retrieval top1 when it is not rejected; include lower-rank
   ↪  candidates
   only when records clearly transfer.

Return JSON only with this schema:
{
  "top1_exp_id": "...",
  "top3_exp_ids": ["...", "...", "..."],
  "ranked_exp_ids": ["..."],
  "current_bottleneck": "short phrase",
  "candidate_judgments": [
    {
      "exp_id": "...",
      "partition": "strong_transfer|plausible_transfer|weak_or_partial|
        reject_or_boundary",
      "applicable_phase": "high|medium|low",
      "chain_evidence": "full_chain|critical_partial|search_only|
        reduced_invalid_only|none",
      "record_support": "strong_positive|positive|weak|
        negative_or_boundary|unclear",
      "hard_reject":
      ↪  "none|wrong_task_family|wrong_destination|wrong_tool|
        wrong_transform|wrong_count|wrong_target_or_analogy_only|wrong_s
        ↪  tage",
      "retrieval_rank_prior":
      ↪  "keep_high|neutral|overridden_by_strong_evidence",
      "rank_decision": "promote|keep|demote|reject",
      "reason": "short evidence-grounded reason"
    }
  ],
  "reason": "short final listwise reason"
}

CURRENT_CASE_AND_EVIDENCE:
{current_case_and_candidate_evidence}
```

For ScienceWorld, ι_{rank} uses the same schema but the serialized fields describe protocol-specific evidence, such as protocol stage, operation, state cue, successful action boundary, final focus or answer action, and wrong-protocol annotations. The output remains a single selected experience e.

Experience-Guided LLM Reasoning for ALFWorld. After Eq. 7 selects e, the experience is injected into the reasoning prompt as optional local guidance.

```
Interact with a household to solve a task. You may take maximum of 20
↪  steps.
Here are two examples.
{task_family_react_examples}

At each turn, follow the same ReAct format as the examples: write a brief
`Think:` line if useful, then write exactly one `Action:` line. Stop
↪  after the
Action line; do not write the next Observation.
```

```
Retrieved decision-memory hints for the next local decision:
{selected_experience}

Use a hint only if its object, receptacle, task phase, and current state match
this task. Do not copy a remembered trajectory as a script; keep following the
ReAct examples and choose the next valid action for the current observation.

Here is the task:
{initial_observation_and_task}
{recent_action_observation_history}
```

Experience-Guided LLM Reasoning for ScienceWorld. ScienceWorld uses a different action space and therefore uses a separate reasoning prompt. The selected experience is still optional local guidance and must be checked against the current scientific subgoal, observation, history, and admissible actions.

```
You are a helpful assistant to do some scientific experiment in an environment.
The available actions are:
open OBJ, close OBJ, activate OBJ, deactivate OBJ, connect OBJ to OBJ,
disconnect OBJ, use OBJ [on OBJ], look around, examine OBJ, look at OBJ,
read OBJ, move OBJ to OBJ, pick up OBJ, pour OBJ into OBJ, mix OBJ,
teleport to LOC, focus on OBJ, wait, wait1.

Here is an example.
{scienceworld_react_example}

Now solve the new task.
{task}
Observation: {current_observation}
{recent_action_observation_history}

Retrieved decision-memory hints are available for the next local decision.
Keep the normal ReAct policy as primary: infer the current scientific subgoal
from the task, observation, history, and admissible actions first. Then use a
memory hint only when its phase, objects/materials, room/state, and target
outcome match this exact local state.
Do not copy a hint as a script. Do not switch to a remembered object, room,
container, or final target unless it is required by the current task and visible
or admissible now. If no hint clearly matches, ignore memory and continue the
baseline ReAct plan.

Retrieved decision-memory hints:
{selected_experience}

Reminder: the current admissible actions are:
{admissible_actions}

Think briefly about the current scientific subgoal. If a memory hint matches
this local state, use it only to choose the next admissible action; otherwise
ignore it. Then output exactly one admissible action.
Thought:
```

Usage Record Generation Prompt ι_{rec}. This prompt implements Eq. 10. After an experience e_i is used or evaluated, ι_{rec} converts the rollout feedback into a usage record $u_{i,j}$.

```
You are writing one usage record for an experience.

Input:
- DECISION_STATE s
- SELECTED_EXPERIENCE e_i
- MEMORY_CONDITIONED_ROLLOUT tau
- MEMORYLESS_OR_BASELINE_ROLLOUT tau^0
- FEEDBACK_LABEL y
- REWARD_DIFFERENCE Delta R

Goal:
Create structured feedback evidence that a future reranker can compare
↪  with a
new query. The record should describe when this experience helped,
↪  failed, or
did not matter.

Rules:
- Do not summarize the whole trajectory.
- Preserve the local decision state in which the experience was used.
- Record the structured fields of the current state phi(s), the
↪  experience
  fields phi(e_i), the realized trajectory summary psi(tau), feedback y,
  ↪  and
  reward difference Delta R.
- Keep concrete mismatch fields when they define transfer boundaries.
- Do not make the final utility judgment for future problems; provide
↪  evidence
  for the later LLM Listwise Reranking prompt.

Return JSON only:
{
  "state_fields": {
    "task_family": "...",
    "target": "...",
    "destination": "...",
    "tool": "...",
    "required_state": "...",
    "count": "...",
    "current_phase": "...",
    "required_chain": ["..."],
    "covered_chain": ["..."]
  },
  "experience_fields": {
    "experience_id": "...",
    "experience_stage": "...",
    "principle": "...",
    "transfer_boundary": "..."
  },
  "trajectory_summary": "...",
  "feedback_label": "...",
  "delta_R": ...,
  "mismatch_annotations": ["..."],
  "record_summary": "..."
}
```