HIERAMEM: EVIDENCE-ANCHORED HIERARCHICAL MEMORY FOR SELF-IMPROVING LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-evolving LLM agents improve over time by accumulating and reusing past experience through memory. However, constructing memory from prior interactions entails a fundamental tension between preserving evidence and abstracting knowledge: complete trajectories retain rich behavioral evidence but introduce substantial noise, whereas abstract memories facilitate reuse but often obscure their evidential basis. To address this tension, we introduce HIERAMEM, an evidence-anchored hierarchical memory framework that bridges complete interaction trajectories and reusable knowledge through compact, traceable evidence anchors. During memory formation, HIERAMEM contrasts related trajectories with different task outcomes to identify behaviorally discriminative action–observation evidence and retain it as evidence anchors. These anchors constrain the abstraction of transferable decision knowledge, forming a three-level hierarchy comprising episodic evidence, evidence anchors, and reusable memory. As experience accumulates, this hierarchy evolves across episodes while keeping reusable knowledge grounded in its supporting anchors. At inference time, the same anchors support memory retrieval at two granularities, serving as global references at the task level and as local addresses at the decision level. Experiments across five benchmarks show that HIERAMEM improves overall agent performance over existing memory systems, demonstrating the effectiveness of evidence anchoring in connecting memory abstraction with reliable reuse.

1 INTRODUCTION

Large language model (LLM) agents have demonstrated strong capabilities in reasoning, planning, and interactive decision making (Yao et al., 2023; Huang et al., 2024). Building on these capabilities, self-evolving LLM agents improve by accumulating and reusing interaction experience, making memory central to continual adaptation (Zhao et al., 2024). Recent work on agent memory has demonstrated the value of experience reuse across embodied interaction (Wang et al., 2024a; Majumder et al., 2023), web navigation (Fu et al., 2024; Wang et al., 2025), software engineering (Ouyang et al., 2026; Mu et al., 2026), and knowledge-intensive reasoning (Zhang et al., 2025a; Cheng et al., 2026).

The effectiveness of agent memory depends on the level of abstraction at which past experience is represented. Existing methods commonly retain interaction trajectories (Zheng et al., 2024; Kagaya et al., 2024) or distill them into higher-level reflections, workflows, and strategies (Shinn et al., 2023; Zhao et al., 2024; Fu et al., 2024; Wang et al., 2025). Complete trajectories preserve the interaction context needed to interpret past decisions, but also retain exploratory steps, redundant operations, and episode-specific details (Hu et al., 2025a; Liang et al., 2026). Abstract memories improve transfer across tasks by removing such specificity (Sarch et al., 2024; Liang et al., 2026), but may also discard evidence that distinguishes when different decisions are appropriate. Memory construction therefore faces a tension between *evidence preservation* and *knowledge abstraction*. For self-evolving agents, this tension directly shapes how reliably memory can be reused and evolved over time. A key requirement is therefore removing details that restrict generalization while preserving the interaction evidence that defines the valid scope of the resulting abstraction.

To address this question, we introduce HIERAMEM, an evidence-anchored hierarchical memory framework for self-evolving LLM agents. The key idea is to preserve the evidence that grounds reusable knowledge while allowing the knowledge itself to remain abstract, and to maintain this

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.

grounding throughout memory formation, evolution, and use. Specifically, HIERAMEM organizes memory into three connected levels: *Episodic Evidence* preserves complete interaction trajectories, *Evidence Anchors* retains correction- and progress-relevant evidence identified from related episodes with different outcomes, and *Reusable Memory* stores transferable decision knowledge grounded in that evidence. Across successive episodes, reusable knowledge is consolidated while preserving its associations with supporting evidence. These same anchors further support memory access at complementary granularities, providing task-level global references and decision-level local guidance.

Our contributions are summarized as follows:

1. We propose HIERAMEM, an evidence-anchored hierarchical memory framework for self-evolving LLM agents. HIERAMEM introduces *Evidence Anchors* as an intermediate representation between complete interaction trajectories and reusable knowledge, preserving behaviorally relevant evidence while supporting progressive knowledge abstraction across episodes.
2. We introduce a dual-granularity memory utilization mechanism that carries retained evidence from memory construction into subsequent reuse. It combines task-level guidance with decision-level retrieval conditioned on the agent's current interaction state, enabling precise memory reuse to adapt to changing decision needs.
3. Experiments across five benchmarks spanning embodied interaction, symbolic planning, and knowledge-intensive reasoning demonstrate consistent improvements over existing memory systems, with HIERAMEM achieving the best or tied-best performance across all evaluated settings.

2 RELATED WORK

Memory Construction and Abstraction. Agent memory construction converts interaction experience into information for reuse in later tasks (Zhang et al., 2025c; Wang et al., 2024b). Some methods retain trajectories as task exemplars, as in Synapse (Zheng et al., 2024), while others abstract experience into reflections, workflows, or conditional guidelines (Shinn et al., 2023; Wang et al., 2025; Fu et al., 2024; Park et al., 2023). To derive reusable guidance, methods such as ExpeL, ReasoningBank, and ReMe draw on both successful and failed experiences (Zhao et al., 2024; Ouyang et al., 2026; Cao et al., 2026). Abstraction supports transfer by reducing episode-specific detail, but may also remove local interaction evidence needed to assess when the resulting guidance applies.

Memory Organization and Evolution. After construction, agent memory must be organized for access and updated as new experience accumulates (Luo et al., 2026). Retrieval-based approaches maintain external stores that provide context for model generation (Lewis et al., 2020; Sarthi et al., 2024; Gutiérrez et al., 2024; Packer et al., 2023; Zhong et al., 2024; Modarressi et al., 2023; 2025; Salama et al., 2025). Within such stores, knowledge can be organized as atomic notes, semantic hierarchies, or graphs (Xu et al., 2025; Sun et al., 2026; Zhang et al., 2025a; Chhikara et al., 2025; Hu et al., 2025a). As new episodes arrive, memory maintenance incorporates additional experience and revises existing knowledge (Zhong et al., 2024; Xu et al., 2025; Chhikara et al., 2025; Cao et al., 2026). Existing approaches support rule updates, adaptive abstraction, or utility-based addition and pruning (Chen et al., 2024; Liang et al., 2026; Zhang et al., 2026). However, these strategies provide limited support for maintaining explicit links to supporting interaction evidence as abstract knowledge is revised or consolidated.

Memory Retrieval and Reuse. During task execution, agents retrieve stored memories to support current decisions (Hu et al., 2025b; Liu et al., 2023). Many methods match the current task goal or query against stored experience (Zheng et al., 2024; Zhao et al., 2024; Ouyang et al., 2026). Others incorporate interaction state or scenario context into retrieval (Fu et al., 2024; Gao et al., 2025; Cao et al., 2026; Wang et al., 2026), or use hierarchical and graph relations to guide memory access (Sun et al., 2026; Zhang et al., 2025a). These strategies improve contextual matching and search organization, but a relevant trajectory or abstract memory may still concern decision conditions that differ from the current situation. Using supporting interaction evidence as an intermediate retrieval target offers a way to assess these conditions before accessing the linked knowledge.

3 PRELIMINARIES

Agent Interaction with External Memory. A ReAct-style LLM agent π_θ solves a sequence of task instances $\{\mathcal{T}_i\}_{i=1}^{N}$, where each task $\mathcal{T}_i$ is specified by a goal g_i. Let $\mathcal{H}_{i-1}$ denote the external memory

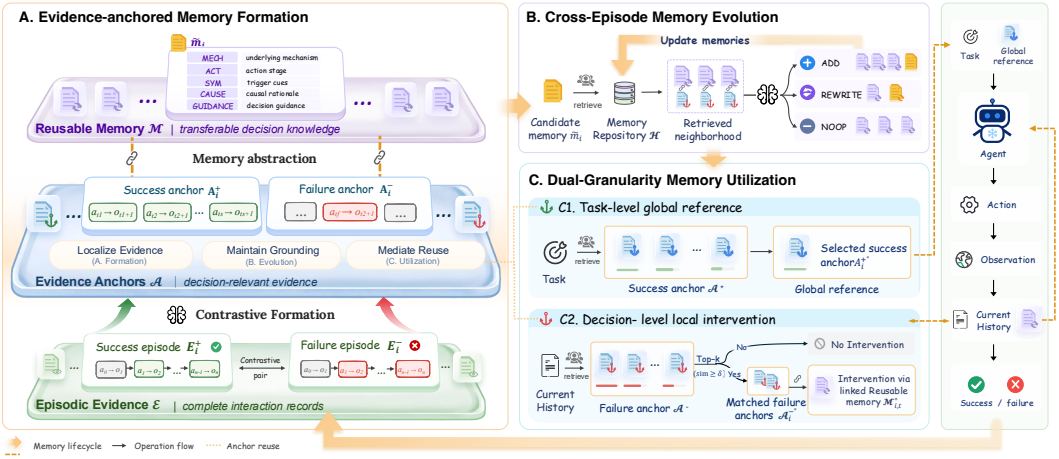


Figure 1: The overview of HIERAMEM across the memory lifecycle. (A) Successful and failed episodes are contrasted to derive decision-relevant Evidence Anchors and Reusable Memory. (B) Reusable knowledge evolves across episodes while maintaining its evidence grounding. (C) The same anchors support task-level references and decision-level access to reusable memory.

accumulated from previously completed episodes and available before episode i. At decision step t, given the current interaction history $h_i^{(t)}$, the memory system retrieves context $c_i^{(t)}$ from $\mathcal{H}_{i-1}$, and the agent selects action as

$$a_i^{(t)} \sim \pi_\theta\left(\cdot \mid g_i,\, h_i^{(t)},\, c_i^{(t)}\right). \tag{1}$$

The resulting trajectory is $\tau_i = (\xi_i^{(1)}, \ldots, \xi_i^{(T_i)})$, where each transition $\xi_i^{(t)} = (o_i^{(t-1)}, a_i^{(t)}, o_i^{(t)})$ records an action and its observable feedback. A completed episode is represented as

$$E_i = (g_i, \tau_i, y_i) \tag{2}$$

where $y_i \in \{0, 1\}$ denotes the terminal task outcome, with 0 and 1 indicating failure and success, respectively. After termination, the memory state is updated from $\mathcal{H}_{i-1}$ to $\mathcal{H}_i$ based on E_i, which is then available to subsequent episodes.

Hierarchical Memory Representation. HIERAMEM organizes accumulated experience into three linked representations: *Episodic Evidence*, *Evidence Anchors*, and *Reusable Memory*. After episode i, the memory state is

$$\mathcal{H}_i = \left(\mathcal{E}_i, \mathcal{A}_i, \mathcal{M}_i; \mathcal{R}_i^{EA}, \mathcal{R}_i^{AM}\right), \tag{3}$$

where $\mathcal{E}_i$, $\mathcal{A}_i$, and $\mathcal{M}_i$ denote the corresponding three representation levels. The cross-level relations $\mathcal{R}_i^{EA} \subseteq \mathcal{E}_i \times \mathcal{A}_i$ and $\mathcal{R}_i^{AM} \subseteq \mathcal{A}_i \times \mathcal{M}_i$ record episode–anchor and anchor–memory associations, respectively. We further write $\mathcal{E}_i = \mathcal{E}_i^- \cup \mathcal{E}_i^+$ and $\mathcal{A}_i = \mathcal{A}_i^- \cup \mathcal{A}_i^+$ for failure/success episodes and their corresponding anchors.

Retrieval over these memory representations is based on semantic similarity. Let q denote the retrieval query, $\mathcal{X}$ a finite candidate set from the relevant memory layer, and $e(\cdot)$ denoting embeddings of fixed textual serializations. The similarity between q and $x \in \mathcal{X}$ is $\mathrm{sim}(q, x) = \frac{e(q)^\top e(x)}{\|e(q)\|_2 \|e(x)\|_2}$, and $\mathrm{TopK}_K(\mathcal{X}; \mathrm{sim}(q, \cdot))$ returns the $\min(K, |\mathcal{X}|)$ highest-scoring candidates.

4 HIERAMEM

4.1 EVIDENCE-ANCHORED MEMORY FORMATION

HIERAMEM forms reusable memory through two coupled contrastive operations: evidence localization and memory abstraction. Evidence localization identifies compact, decision-relevant *Evidence Anchors* through success–failure comparison of the complete interaction records in *Episodic Evidence*. Memory abstraction distills transferable *Reusable Memory* from this comparison, with explicit links to the anchors that preserve its supporting evidence. Both operations are grounded in the same contrastive episode pair. The formation process is illustrated in Figure 1A.

Memory formation in HIERAMEM is triggered by a failed episode. The episode records the actions and observations of an unsuccessful attempt. However, the failed episode alone provides limited evidence for identifying which local decisions are relevant to correction and how they should be revised. HIERAMEM pairs it with a successful episode from a related task as a contrastive reference.

Each completed episode is first added to *Episodic Evidence*. For a failed episode E_i^-, HIERAMEM selects the previously successful episode with the most similar goal. Let $g(E)$ denote the goal of episode E. The reference episode is

$$\widehat{E}_i^+ = \arg \max_{E \in \mathcal{E}_{i-1}^+} \mathrm{sim}\big(g_i, g(E)\big). \tag{4}$$

If no successful episode is available, the failed episode is retained in *Episodic Evidence* and used for cold-start formation, from which a failure anchor and its candidate memory are derived.

The contrastive pair $(E_i^-, \widehat{E}_i^+)$ supports the localization of two *Evidence Anchors* that preserve behaviorally relevant evidence from the comparison. The failure anchor A_i^- identifies a local decision relevant to correction and retains its surrounding actions and observations as contextual evidence. The success anchor A_i^+ retains sparse, temporally ordered steps that capture effective task progress. These steps need not form a contiguous window or correspond step by step to the failure evidence.

However, the retained evidence remains specific to individual episodes and is difficult to reuse directly as decision knowledge. Memory abstraction distills the contrastive comparison into *Reusable Memory*, which captures a transferable local decision pattern and state-aware corrective guidance, with explicit links to the supporting *Evidence Anchors*. Because the resulting knowledge has not yet been reconciled with existing memory, it is represented as a candidate

$$\widetilde{m}_i = (z_i, u_i), \tag{5}$$

where z_i describes the decision pattern through $MECH_i$, ACT_i, SYM_i, and $CAUSE_i$. These fields represent the decision mechanism, decision stage, observable runtime trigger cues, and a decisive state contrast, respectively. The component u_i specifies reusable corrective guidance associated with the failure anchor and supported by the comparison.

Evidence localization and guidance abstraction are jointly performed conditioned on the same success–failure pair. We represent the overall contrastive formation procedure as

$$(A_i^-, A_i^+, \widetilde{m}_i) = \mathcal{C}_{\mathrm{LLM}}^{\mathrm{form}}\big(E_i^-, \widehat{E}_i^+; \mathcal{P}_f\big). \tag{6}$$

where $\mathcal{C}_{\mathrm{LLM}}^{\mathrm{form}}$ denotes a prompt-conditioned LLM procedure that localizes supporting interaction evidence and abstracts reusable decision guidance from the same contrastive pair. If the comparison provides insufficient support for reusable knowledge, the procedure returns no candidate. Details of the formation prompt are provided in Appendix D.

4.2 CROSS-EPISODE MEMORY EVOLUTION

A candidate derived from a single comparison may introduce new knowledge or overlap with, refine, or restate existing knowledge. Cross-episode evolution therefore compares the candidate with related *Reusable Memory* and their supporting evidence to determine whether to add the candidate, consolidate and revise existing knowledge, or leave the memory unchanged.

For $\widetilde{m}_i$, HIERAMEM uses the pattern description z_i to retrieve a neighborhood $\mathcal{N}_i$ of existing *Reusable Memory*. Let $z(m)$ denote the pattern description associated with memory m. The retrieved neighborhood is

$$\mathcal{N}_i = \mathrm{TopK}_e\left(\mathcal{M}_{i-1}; m \mapsto \mathrm{sim}\big(z_i, z(m)\big)\right). \tag{7}$$

where K_e denotes the retrieval budget for memory evolution. Semantic similarity alone is insufficient to determine whether memories should be consolidated, so HIERAMEM compares their associated failure anchors to assess whether the reusable memories address the same local decision problem under comparable interaction conditions.

Based on this evidence-aware comparison, HIERAMEM determines the evolution operation:

$$(d_i, \mathcal{S}_i, \widehat{m}_i) = \mathcal{C}_{\mathrm{LLM}}^{\mathrm{evolve}}\Big((\widetilde{m}_i, A_i^-), \{(m, \mathcal{A}_{i-1}^-(m)) \mid m \in \mathcal{N}_i\}; \mathcal{P}_e\Big), \tag{8}$$

where $\mathcal{C}_{\text{LLM}}^{\text{evolve}}$ denotes the prompt-conditioned LLM operator with evolution prompt $\mathcal{P}_e$ (see Appendix D), and $\mathcal{A}_{i-1}^{-}(m)$ denotes the supporting failure-side anchors of m presented in the current evolution call, satisfying

$$\mathcal{A}_{i-1}^{-}(m) \subseteq \left\{ A^- \in \mathcal{A}_{i-1}^{-} \mid (A^-, m) \in \mathcal{R}_{i-1}^{AM} \right\}. \tag{9}$$

The variable $d_i \in \{\text{ADD}, \text{REWRITE}, \text{NOOP}\}$ denotes the evolution operation: ADD retains $\widetilde{m}_i$ as a new memory, REWRITE replaces the memories in $\mathcal{S}_i$ with the revised memory $\widehat{m}_i$, and NOOP leaves the current *Reusable Memory* unchanged. If no related memory is retrieved, the candidate is added directly without cross-memory comparison.

The selected operation is then applied to the hierarchical memory state. The completed episode is always archived in *Episodic Evidence*, while the upper two levels are updated according to the evolution decision. The resulting update to *Reusable Memory* is

$$\mathcal{M}_i = \begin{cases} \mathcal{M}_{i-1} \cup \{\widetilde{m}_i\}, & d_i = \text{ADD}, \\ (\mathcal{M}_{i-1} \setminus \mathcal{S}_i) \cup \{\widehat{m}_i\}, & d_i = \text{REWRITE}, \\ \mathcal{M}_{i-1}, & d_i = \text{NOOP}. \end{cases} \tag{10}$$

When an update is accepted by ADD or REWRITE, its supporting anchors are incorporated into *Evidence Anchors*, with their episode provenance recorded by $\mathcal{R}^{EA}$. Under ADD, the new anchors are linked to $\widetilde{m}_i$. Under REWRITE, the new anchors are linked to $\widehat{m}_i$, while the historical anchor–memory associations previously linked to memories in $\mathcal{S}_i$ are redirected to $\widehat{m}_i$. Under NOOP, neither the candidate memory nor its new anchors are incorporated into the upper two memory levels, as the candidate is covered by existing memory or lacks sufficient support for reuse. These updates yield the new memory state $\mathcal{H}_i$ while preserving the accumulated evidence associations across episodes.

4.3 DUAL-GRANULARITY MEMORY UTILIZATION

When HIERAMEM encounters a new task, the memory hierarchy constructed through formation and evolution provides prior experience to guide the ongoing interaction. Because task-level progression and local decision making require memory at different granularities, HIERAMEM adopts a dual-granularity utilization strategy. Success anchors provide a global reference for successful task progress, whereas failure anchors serve as local retrieval cues for accessing *Reusable Memory* relevant to the current interaction context.

Task-level global reference. At the beginning of task $\mathcal{T}_i$, HIERAMEM compares its goal g_i with the goals of historical success anchors and selects the most relevant one as the task-level reference:

$$A_i^{+*} = \arg \max_{A^+ \in \mathcal{A}_{i-1}^{+}} \text{sim}\big(g_i, g(A^+)\big), \tag{11}$$

where $g(A^+)$ denotes the task goal associated with A^+. The selected anchor provides a sparse, temporally ordered trace of successful task progress and remains fixed throughout the episode. If no success anchor is available, no task-level reference is provided.

Decision-level local intervention. As the interaction unfolds, new actions and observations continually change the local context, requiring relevant memory to be selected dynamically. Failure anchors preserve the local interaction evidence associated with *Reusable Memory*, grounding abstract guidance in the concrete conditions under which it applies. HIERAMEM therefore periodically retrieves failure anchors that match the current interaction context and follows their associations to access the corresponding *Reusable Memory*.

To identify relevant anchors, HIERAMEM constructs a query from the task goal g_i and the most recent w transitions. At decision step t, the interaction window is $h_i^{(w,t)} = \big(\xi_i^{(s)}\big)_{s=\max(1,t-w)}^{t-1}$, and the corresponding query is $q_i^{(t)} = \text{Serialize}\big(g_i, h_i^{(w,t)}\big)$. HIERAMEM then retrieves the highest-scoring historical failure anchors whose similarity exceeds a threshold:

$$\mathcal{A}_i^{-*(t)} = \text{TopK}_{K_{\text{mem}}}\left(\left\{ A^- \in \mathcal{A}_{i-1}^{-} : \text{sim}\big(q_i^{(t)}, A^-\big) \geq \delta \right\}; A^- \mapsto \text{sim}\big(q_i^{(t)}, A^-\big) \right), \tag{12}$$

where δ denotes the minimum similarity threshold and K_{mem} the maximum number of retrieved anchors. If no anchor satisfies the threshold, no decision-level intervention is provided.

The retrieved failure anchors then serve as addresses into *Reusable Memory*. Following the anchor–memory associations maintained by $\mathcal{R}^{AM}$ during memory evolution, HIERAMEM accesses the memories linked to the retrieved anchors:

$$\mathcal{M}_i^{*(t)} = \left\{ m \in \mathcal{M}_{i-1} : \exists A^- \in \mathcal{A}_i^{-*(t)},\ (A^-, m) \in \mathcal{R}_{i-1}^{AM} \right\}. \tag{13}$$

For each memory $m \in \mathcal{M}_i^{*(t)}$, its observable runtime trigger cues SYM_m and reusable decision guidance u_m are included in the agent's memory context. The former characterizes the local conditions associated with the memory, while the latter specifies the corresponding corrective guidance. Because memory evolution may associate multiple failure anchors with the same consolidated memory, different retrieved anchors can map to the same m. Consequently, $\mathcal{M}_i^{*(t)}$ may contain fewer than K_{mem} distinct memories. No additional ranking over the retrieved memory content is performed after this mapping.

Finally, the task-level reference A_i^{+*} and the trigger cues and guidance from $\mathcal{M}_i^{*(t)}$ together form the external memory context $c_i^{(t)}$ in Equation (1). The former provides a global reference for task progression, whereas the latter supplies context-specific guidance for local decision making.

5 EXPERIMENTS

In this section, we evaluate HIERAMEM from three perspectives. **(RQ1)** How does HIERAMEM compare with existing memory systems across tasks and backbones? **(RQ2)** How effective are evidence-anchored formation and cross-episode evolution? **(RQ3)** How do Evidence Anchors support memory retrieval and reuse?

5.1 EXPERIMENTAL SETUP

Benchmarks. We evaluate HIERAMEM on five benchmarks spanning three task families: embodied interaction (ALFWorld (Shridhar et al., 2021) and ScienceWorld (Wang et al., 2022)), symbolic planning (PDDL from AgentBoard (Ma et al., 2024)), and knowledge reasoning (HotpotQA (Yang et al., 2018) and FEVER (Thorne et al., 2018)). Detailed benchmark settings and evaluation metrics are provided in Appendix A.1.

Baselines. We compare HIERAMEM with No Memory and five representative experiential-memory methods: Voyager (Wang et al., 2024a), MemoryBank (Zhong et al., 2024), Generative Agents (Park et al., 2023), Agent Workflow Memory (AWM) (Wang et al., 2025), and ExpeL (Zhao et al., 2024). All methods use the same task agent, interfaces, and evaluation settings wherever applicable, while retaining their original memory mechanisms. Details of baselines are provided in Section A.2.

LLM Backbones and Experimental Protocol. We evaluate HIERAMEM with two open-weight backbones, Qwen3-8B and Qwen3-14B (Yang et al., 2025), and one proprietary backbone, GPT-4o-mini. Semantic retrieval is performed with `qwen3-embedding:4b` (Zhang et al., 2025b). The Qwen backbones and the embedding model are deployed locally through Ollama[1], whereas GPT-4o-mini is accessed via the OpenAI API. The same backbone is used for both task execution and all LLM-based memory operations. Reported results are averaged over three independent runs. Unless otherwise specified, ablations and further analyses are conducted with Qwen3-14B. Implementation details are provided in Section A.3.

Parameter Configurations. The main configurable parameters of HIERAMEM are associated with memory retrieval. The context window w specifies the local interaction context, while K_{mem} and δ control failure-anchor retrieval in eq. (12). Task-level reference selection follows eq. (11) without additional parameters. We use the same parameter configuration across all benchmarks and backbones. Details are reported in Section A.3.

5.2 MAIN RESULTS (RQ1)

Table 1 reports the performance of different memory systems across five benchmarks and three LLM backbones. We summarize two main observations.

[1] https://github.com/ollama/ollama

Table 1: Performance comparison of HIERAMEM with existing memory systems across five benchmarks. Numbers following ↑/↓ indicate absolute changes over the corresponding No-memory baseline. The best and runner-up results are highlighted.

Model	Memory	ALFWorld	SciWorld	PDDL	HotpotQA	FEVER	Avg.
GPT-4o-mini	No-memory	42.54 ↑0.00	40.00 ↑0.00	16.67 ↑0.00	26.26 ↑0.00	55.56 ↑0.00	36.21 ↑0.00
	Voyager	47.01 ↑4.47	45.56 ↑5.56	18.33 ↑1.66	24.00 ↓2.26	56.57 ↑1.01	38.29 ↑2.08
	MemoryBank	43.89 ↑1.35	43.89 ↑3.89	15.00 ↓1.67	24.24 ↓2.02	57.58 ↑2.02	36.92 ↑0.71
	Generative	44.78 ↑2.24	50.00 ↑10.00	16.11 ↓0.56	25.25 ↓1.01	58.59 ↑3.03	38.95 ↑2.74
	AWM	34.33 ↓8.21	44.44 ↑4.44	13.33 ↓3.34	29.29 ↑3.03	51.52 ↓4.04	34.58 ↓1.63
	ExpeL	46.27 ↑3.73	46.30 ↑6.30	16.67 ↑0.00	30.64 ↑4.38	58.67 ↑3.11	39.71 ↑3.50
	HIERAMEM (Ours)	**47.76** ↑5.22	**51.11** ↑11.11	**19.17** ↑2.50	**31.31** ↑5.05	**59.60** ↑4.04	**41.79** ↑5.58
Qwen3-8B	No-memory	50.75 ↑0.00	37.78 ↑0.00	18.33 ↑0.00	28.33 ↑0.00	60.61 ↑0.00	39.16 ↑0.00
	Voyager	51.87 ↑1.12	41.11 ↑3.33	17.50 ↓0.83	33.00 ↑4.67	63.64 ↑3.03	41.42 ↑2.26
	MemoryBank	52.25 ↑1.50	42.59 ↑4.81	20.83 ↑2.50	35.35 ↑7.02	66.67 ↑6.06	43.54 ↑4.38
	Generative	52.49 ↑1.74	43.70 ↑5.92	18.89 ↑0.56	27.27 ↓1.06	65.66 ↑5.05	41.60 ↑2.44
	AWM	28.36 ↓22.39	40.37 ↑2.59	20.00 ↑1.67	32.67 ↑4.34	63.30 ↑2.69	36.94 ↓2.22
	ExpeL	55.22 ↑4.47	44.44 ↑6.66	23.33 ↑5.00	35.02 ↑6.69	64.65 ↑4.04	44.53 ↑5.37
	HIERAMEM (Ours)	**55.47** ↑4.72	**45.19** ↑7.41	**24.58** ↑6.25	**36.03** ↑7.70	**69.36** ↑8.75	**46.13** ↑6.97
Qwen3-14B	No-memory	57.46 ↑0.00	44.07 ↑0.00	27.92 ↑0.00	29.97 ↑0.00	64.65 ↑0.00	44.81 ↑0.00
	Voyager	66.42 ↑8.96	43.33 ↓0.74	27.78 ↓0.14	34.68 ↑4.71	64.67 ↑0.02	47.38 ↑2.57
	MemoryBank	56.72 ↓0.74	44.81 ↑0.74	30.83 ↑2.91	37.04 ↑7.07	65.66 ↑1.01	47.01 ↑2.20
	Generative	65.67 ↑8.21	48.52 ↑4.45	28.89 ↑0.97	28.28 ↓1.69	66.67 ↑2.02	47.61 ↑2.80
	AWM	60.70 ↑3.24	46.67 ↑2.60	30.42 ↑2.50	30.98 ↑1.01	64.31 ↓0.34	46.62 ↑1.81
	ExpeL	65.92 ↑8.46	45.56 ↑1.49	29.44 ↑1.52	35.69 ↑5.72	66.67 ↑2.02	48.66 ↑3.85
	HIERAMEM (Ours)	**70.90** ↑13.44	**53.33** ↑9.26	**32.92** ↑5.00	**38.72** ↑8.75	**67.34** ↑2.69	**52.64** ↑7.83

① HIERAMEM consistently improves performance across all evaluated benchmarks and LLM backbones. As shown in Table 1, HIERAMEM achieves the highest score in all fifteen backbone–benchmark settings and improves over No-memory in every case. It raises the average score by 5.58 points with GPT-4o-mini, 6.97 points with Qwen3-8B, and 7.83 points with Qwen3-14B. It also exceeds ExpeL, the strongest competing method on average under each backbone, by 2.08, 1.60, and 3.98 points, respectively. The gains are particularly pronounced on SciWorld (+11.11 points) with GPT-4o-mini, and on ALFWorld (+13.44 points) and HotpotQA (+8.75 points) with Qwen3-14B.

② Existing memory systems vary substantially in the consistency of their gains across tasks. Several competing methods improve some settings but degrade others. With Qwen3-14B, Voyager gains 8.96 points on ALFWorld but loses 0.74 and 0.14 points on SciWorld and PDDL, respectively, while MemoryBank improves PDDL and HotpotQA by 2.91 and 7.07 points but reduces ALFWorld performance by 0.74 points. Similar reversals also appear with GPT-4o-mini, indicating that existing memory representations do not transfer uniformly across tasks. ExpeL is the only competing method that does not degrade performance in any setting. These consistent gains may be related to its use of contrastive abstraction, which compares successful and failed experiences to derive reusable guidance. HIERAMEM likewise exploits contrastive experience, but further associates each reusable memory with the interaction evidence that supports it. This pattern is consistent with our motivation to retain evidence for assessing when reusable knowledge applies to the current decision context.

5.3 MEMORY FORMATION AND EVOLUTION (RQ2)

③ Evidence-anchored memory formation and cross-episode memory evolution both benefit task performance. Figure 2 evaluates these two components through controlled ablations. For formation, we replace evidence-anchored formation in Equation (6) with *Direct Memory Abstraction*, which abstracts Reusable Memory directly from the same contrasted episodes without grounding it in the identified Evidence Anchors. For evolution, we retain evidence-anchored formation but replace the memory evolution defined in Equation (8) with *Append-only* accumulation. Both variants reduce perfor-

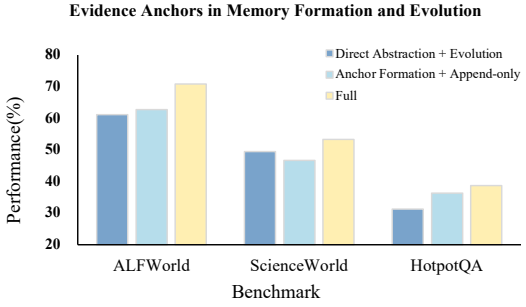


Figure 2: Effect of evidence-anchor formation and cross-episode evolution on task performance.

mance across all three benchmarks. Direct Memory Abstraction causes drops of 9.71, 3.84, and 7.41 points on ALFWorld, ScienceWorld, and HotpotQA, respectively. Append-only causes corresponding drops of 8.21, 6.66, and 2.36 points. Together, these results show that HIERAMEM achieves better task performance through evidence-anchored formation and cross-episode evolution.

Table 2: Effects of Evidence Anchors on task- and decision-level memory utilization. The non-ablated utilization level is kept identical to the full HIERAMEM.

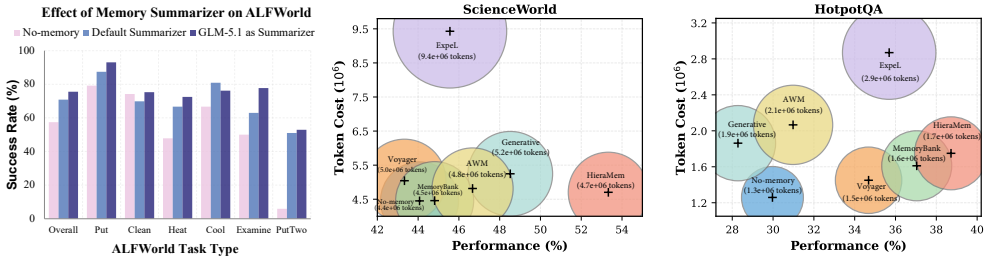
Utilization Level	Reference / Retrieval	ALFWorld	ScienceWorld	HotpotQA
Task-level Reference	Full Trajectory	65.67	47.78	31.31
	None	68.66	51.11	32.32
Decision-level Retrieval	Full Trajectory	66.42	52.22	34.34
	None	60.45	48.89	30.30
Full HIERAMEM		70.90	53.33	38.72

5.4 MEMORY ACCESS AND REUSE (RQ3)

Table 2 reports the effects of Evidence Anchors on task- and decision-level references for memory utilization. We summarize two key findings below.

④ **Success-side evidence provides a more effective task-level reference than complete trajectories.** Removing the success-side anchor A^+ in Equation (11), while retaining decision-level reference, decreases performance by 2.24, 2.22, and 6.40 points on ALFWorld, ScienceWorld, and HotpotQA, respectively. Replacing A^+ with the complete matched successful trajectory causes larger drops of 5.23, 5.55, and 7.41 points. The full-trajectory variant even underperforms the no-reference variant by 2.99, 3.33, and 1.01 points. This shows that simply exposing more successful interaction history does not yield better task-level guidance. Selectively retaining progress-relevant evidence provides a more effective reference than carrying the complete trajectory into subsequent reuse.

⑤ **Anchor-mediated retrieval provides more effective decision-level reuse than full-trajectory access.** Replacing anchor-mediated retrieval defined in Equation (12) with *Full-Trajectory Retrieval* reduces performance by 4.48, 1.11, and 4.38 points on ALFWorld, ScienceWorld, and HotpotQA, respectively. Removing decision-level retrieval while retaining only the global A^+ reference causes larger drops of 10.45, 4.44, and 8.42 points. These results show that complete trajectories retain useful decision-level information, but are less effective than anchor-mediated access. This supports using A^- as a local address for accessing the Reusable Memory associated with the current decision context rather than reasoning over an entire past interaction.



(a) Effect of the memory summarizer. (b) Performance versus token cost across different memory methods.

Figure 3: Further analysis of HIERAMEM.

5.5 FURTHER ANALYSIS

Memory Summarizer Analysis. We further examine whether HIERAMEM depends on the LLM used for memory summarization by replacing the default summarizer with GLM-5.1[2] while keeping Qwen3-14B for task execution. As shown in Figure 3a, the GLM-based variant improves the overall success rate from 70.90 to 75.62, with gains across most ALFWorld task types. This result indicates that the effectiveness of HIERAMEM is not tied to a particular summarization model and can further benefit from stronger memory summarization.

[2] https://bigmodel.cn/

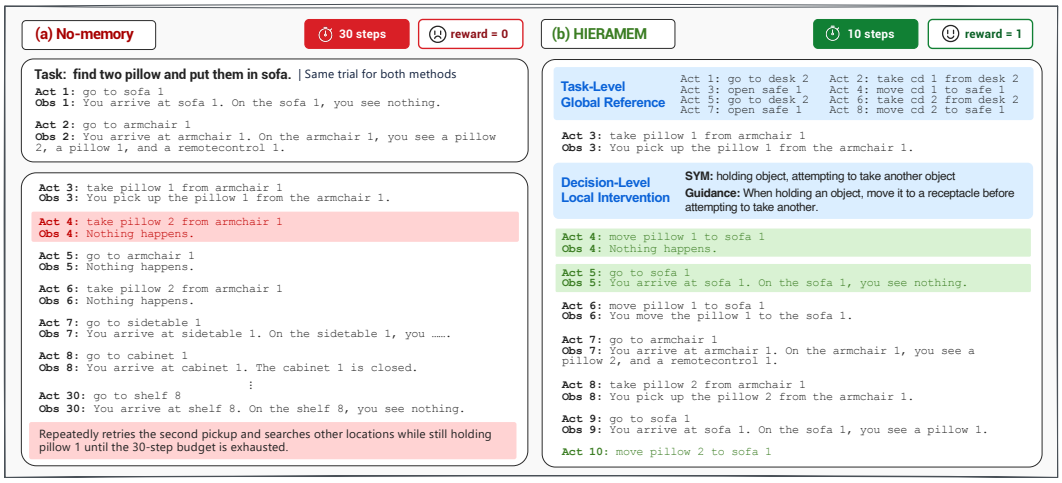


Figure 4: Case study of HIERAMEM on an ALFWorld task.

Token Cost Analysis. We examine the performance–token cost trade-off of different memory methods on ScienceWorld and HotpotQA, as shown in Figure 3b and Figure 6. Token cost is the total number of input and output tokens across all LLM calls in a complete evaluation run. HIERAMEM achieves the highest performance on both benchmarks while maintaining token consumption comparable to existing memory-based baselines. Specifically, it uses approximately 4.7M tokens on ScienceWorld and 1.7M on HotpotQA, both within the range of competing memory methods. These results indicate a favorable trade-off between performance and token cost for HIERAMEM.

5.6 CASE STUDY

Figure 4 illustrates how *Evidence Anchors* make past experience usable during execution in ALFWorld. Both agents reach the same state with *pillow 1* held and *pillow 2* still on the armchair. The *No-memory* agent attempts to take the second pillow, receives "Nothing happens," and subsequently repeats invalid actions and drifts to unrelated locations until the 30-step budget is exhausted. In contrast, HIERAMEM uses a success-side anchor as a global reference for task progress. As the agent's interaction state evolves, it retrieves a matching failure-side anchor for the potentially erroneous pickup and accesses the corresponding Reusable Memory, which guides the agent to place *pillow 1* before taking *pillow 2*. Overall, HIERAMEM retains *Evidence Anchors* as a bridge between interaction evidence and abstracted knowledge, supporting evidence-grounded memory reuse during execution.

6 CONCLUSION AND LIMITATIONS

In this paper, we focus on a recurring limitation in memory systems for self-evolving LLM agents, where reusable knowledge is often abstracted from experience without retaining the interaction evidence needed for subsequent evolution and reuse. To address this limitation, we propose HIERAMEM, an evidence-anchored hierarchical memory framework that maintains the connection between interaction evidence and reusable knowledge throughout the memory lifecycle. Through Evidence Anchors, HIERAMEM grounds memory formation and evolution in supporting evidence and enables task- and decision-level memory utilization. Across all evaluated benchmarks and LLM backbones, HIERAMEM achieves the best performance in all evaluated settings, with consistent improvements over the corresponding No-memory baselines. Ablation studies further show the benefits of *Evidence Anchors* across memory formation, evolution, and utilization. **Limitations.** Although HIERAMEM achieves consistent gains across five benchmarks, several questions remain for future study. Our sensitivity analysis suggests that the preferred retrieval budget varies across tasks, motivating adaptive retrieval beyond the shared configuration used in our experiments. Additionally, HIERAMEM relies on outcome labels to form success–failure pairs for memory formation. In open-ended tasks, noisy labels from user feedback or model evaluation may produce inappropriate pairs, affecting evidence localization and memory abstraction. Future work could improve outcome evaluation and use downstream utility to validate and refine stored memories.

AI USE STATEMENT

LLMs were used for language polishing and formatting, as well as routine assistance with proofreading, literature search queries, and code completion suggestions. The research ideas, experimental design, implementation, and analysis were entirely conducted by the authors. All AI-assisted outputs were manually reviewed and approved by the authors. The authors take full responsibility for all ideas, methods, and claims presented in this paper.

ETHICS STATEMENT

All datasets used in this work are publicly available. Our experiments do not involve human subjects or the collection of personal or sensitive user data. As persistent memory may raise privacy or safety concerns in real-world deployments, our evaluation is restricted to controlled research benchmarks.

REPRODUCIBILITY STATEMENT

We provide detailed information for reproducing our results in the main paper and appendix, including benchmark settings, evaluation metrics, baseline configurations, implementation details, parameter settings, prompts for LLM-based memory operations, and hardware configurations. The evaluated LLM backbones, embedding model, and relevant software dependencies are also specified. We evaluate HIERAMEM with both open-weight and proprietary LLM backbones. All methods are evaluated using the same task interface, evaluation protocol, and task order within each run.

REFERENCES

Zouying Cao, Jiaji Deng, Li Yu, Weikang Zhou, Zhaoyang Liu, Bolin Ding, and Hai Zhao. Remember me, refine me: A dynamic procedural memory framework for experience-driven agent evolution. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 16803–16822, 2026. doi: 10.18653/v1/2026.findings-acl.829.

Minghao Chen, Yihang Li, Yanting Yang, Shiyu Yu, Binbin Lin, and Xiaofei He. AutoManual: Constructing instruction manuals by LLM agents via interactive environmental learning. 37, 2024. doi: 10.52202/079017-0019.

Zihao Cheng, Zeming Liu, Yingyu Shan, Xinyi Wang, Xiangrong Zhu, Yunpu Ma, Hongru Wang, Yuhang Guo, Wei Lin, and Yunhong Wang. Mem^{2}Evolve: Towards self-evolving agents via co-evolutionary capability expansion and experience distillation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 20784–20831, San Diego, California, United States, 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.952.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. 413:2993–3000, 2025. doi: 10.3233/FAIA251160.

Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. AutoGuide: Automated generation and selection of context-aware guidelines for large language model agents. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Pengyu Gao, Jinming Zhao, Xinyue Chen, and Yilin Long. An efficient context-dependent memory framework for LLM-centric agents. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: Industry Track)*, pp. 1055–1069, 2025. doi: 10.18653/v1/2025.naacl-industry.80.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. HippoRAG: Neurobiologically inspired long-term memory for large language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. HiAgent: Hierarchical working memory management for solving long-horizon agent tasks with large language model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pp. 32779–32798, 2025a.

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, Senjie Jin, Jiejun Tan, Jiongnan Liu, Zeyu Zhang, Zhongxiang Sun, Yutao Zhu, Hao Sun, Boci Peng, Zhenrong Cheng, Xuanbo Fan, Jiaxin Guo, Xinlei Yu, Zhenhong Zhou, Zewen Hu, Jiahao Huo, Junhao Wang, Yuwei Niu, Yu Wang, Zhenfei Yin, Xiaobin Hu, Yue Liao, Qiankun Li, Kun Wang, Wangchunshu Zhou, Yixin Liu, Dawei Cheng, Qi Zhang, Tao Gui, Shirui Pan, Yan Zhang, Philip Torr, Zhicheng Dou, Ji-Rong Wen, Xuanjing Huang, Yu-Gang Jiang, and Shuicheng Yan. Memory in the age of AI agents. *arXiv preprint arXiv:2512.13564*, 2025b.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. Understanding the planning of LLM agents: A survey. *arXiv preprint arXiv:2402.02716*, 2024.

Tomoyuki Kagaya, Thong Jing Yuan, Yuxuan Lou, Jayashree Karlekar, Sugiri Pranata, Akira Kinose, Koki Oguri, Felix Wick, and Yang You. RAP: Retrieval-augmented planning with contextual memory for multimodal LLM agents. *arXiv preprint arXiv:2402.03610*, 2024.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Sirui Liang, Pengfei Cao, Jian Zhao, Wenhao Teng, Xiangwen Liao, Jun Zhao, and Kang Liu. Learning how to remember: A meta-cognitive management method for structured and transferable agent memory. pp. 30733–30753, 2026. doi: 10.18653/v1/2026.findings-acl.1535.

Lei Liu, Xiaoyan Yang, Yue Shen, Binbin Hu, Zhiqiang Zhang, Jinjie Gu, and Guannan Zhang. Think-in-memory: Recalling and post-thinking enable LLMs with long-term memory. *arXiv preprint arXiv:2311.08719*, 2023.

Jinghao Luo, Yuchen Tian, Chuxue Cao, Ziyang Luo, Hongzhan Lin, Kaixin Li, Chuyi Kong, Ruichao Yang, and Jing Ma. From storage to experience: A survey on the evolution of LLM agent memory mechanisms. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 41622–41652. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.findings-acl.2069.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. AgentBoard: An analytical evaluation board of multi-turn LLM agents. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-2365.

Bodhisattwa Prasad Majumder, Bhavana Dalvi Mishra, Peter Jansen, Oyvind Tafjord, Niket Tandon, Li Zhang, Chris Callison-Burch, and Peter Clark. CLIN: A continually learning language agent for rapid task adaptation and generalization. *arXiv preprint arXiv:2310.10134*, 2023.

Ali Modarressi, Ayyoob Imani, Mohsen Fayyaz, and Hinrich Schütze. RET-LLM: Towards a general read-write memory for large language models. *arXiv preprint arXiv:2305.14322*, 2023.

Ali Modarressi, Abdullatif Köksal, Ayyoob Imani, Mohsen Fayyaz, and Hinrich Schütze. MemLLM: Finetuning LLMs to use explicit read-write memory. *Transactions on Machine Learning Research*, 2025.

Fangwen Mu, Junjie Wang, Lin Shi, Song Wang, Shoubin Li, and Qing Wang. ExpeRepair: Dual-memory enhanced LLM-based repository-level program repair. *Proceedings of the ACM on Software Engineering*, 3(FSE):3954–3976, 2026. doi: 10.1145/3808181.

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.

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*, 2023. doi: 10.1145/3586183.3606763.

Rana Salama, Jason Cai, Michelle Yuan, Anna Currey, Monica Sunkara, Yi Zhang, and Yassine Benajiba. MemInsight: Autonomous memory augmentation for LLM agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 33136–33152. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.1683.

Gabriel Sarch, Lawrence Jang, Michael J. Tarr, William W. Cohen, Kenneth Marino, and Katerina Fragkiadaki. VLM agents generate their own memories: Distilling experience into embodied programs of thought. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. RAPTOR: Recursive abstractive processing for tree-organized retrieval. In *International Conference on Learning Representations*, 2024.

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

Haoran Sun, Shaoning Zeng, and Bob Zhang. H-MEM: Hierarchical memory for high-efficiency long-term reasoning in LLM agents. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 341–350, 2026. doi: 10.18653/v1/2026.eacl-long.15.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: A large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 809–819, 2018.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024a. ISSN 2835-8856.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024b. doi: 10.1007/s11704-024-40231-1.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Cote, 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*, 2022.

Tong Wang, Pei Xu, Shiyue Cao, Likun Yang, Daipeng Li, Jianbin Jiao, and Kaiqi Huang. SAMem: State-aware memory as a fine-grained memory for LLM agents in decision-making. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 14691–14710, 2026. doi: 10.18653/v1/2026.findings-acl.722.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

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.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, 2018.

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*, 2023.

Guibin Zhang, Muxin Fu, Kun Wang, Frank Wan, Miao Yu, and Shuicheng Yan. G-Memory: Tracing hierarchical memory for multi-agent systems. In *Advances in Neural Information Processing Systems*, volume 38, 2025a.

Haozhen Zhang, Quanyu Long, Jianzhu Bao, Tao Feng, Weizhi Zhang, Haodong Yue, and Wenya Wang. MemSkill: Learning and evolving memory skills for self-evolving agents. *arXiv preprint arXiv:2602.02474*, 2026.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 Embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025b.

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):155:1–155:47, 2025c. doi: 10.1145/3748302.

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.

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

A Experimental Details

A.1 Benchmark Descriptions

We evaluate all methods on five benchmarks spanning three task families: embodied interaction, symbolic planning, and knowledge reasoning. Table 3 summarizes the evaluation setting.

Table 3: Summary of the benchmarks used in our experiments.

Task Family	Benchmark	Metric	Description
Embodied Interaction	ALFWorld[3]	Success Rate	Text-based household tasks requiring navigation and object interaction.
	ScienceWorld[4]	Success Rate	Interactive science tasks requiring experimentation and procedural reasoning.
Symbolic Planning	PDDL[5]	Success Rate	Symbolic planning tasks requiring valid action sequences to reach target goals.
Knowledge Reasoning	HotpotQA[6]	Exact Match	Multi-hop question answering requiring retrieval and synthesis of supporting evidence.
	FEVER[7]	Accuracy	Fact verification requiring evidence retrieval to assess natural-language claims.

Embodied Interaction. ALFWorld (Shridhar et al., 2021) is a text-based embodied benchmark centered on household manipulation tasks. Given a natural-language goal, the agent navigates the environment and performs actions such as picking, placing, cleaning, heating, cooling, and examining objects. ScienceWorld (Wang et al., 2022) extends interactive text-based evaluation to scientific tasks, where the agent navigates rooms, manipulates objects, and carries out multi-step procedures involving scientific concepts and experiments.

Symbolic Planning. We use the PDDL environment from AgentBoard (Ma et al., 2024). Each task specifies an initial symbolic state and a target goal, and the agent must execute a valid sequence of actions that satisfies the goal under explicit symbolic constraints.

Knowledge Reasoning. HotpotQA (Yang et al., 2018) evaluates multi-hop question answering, where the agent must combine evidence from multiple sources before producing a final answer. FEVER (Thorne et al., 2018) evaluates fact verification, where the agent determines whether a claim is supported, refuted, or lacks sufficient evidence.

A.2 Baseline Details

We compare HieraMem with a no-memory agent and five representative experiential-memory methods. For methods originally developed for specific domains, we adapt their memory interfaces to our evaluation setting while preserving their core memory construction, update, and retrieval mechanisms.

- **No Memory:** The No Memory baseline uses the same task agent without cross-episode memory. The agent retains the interaction history within the current episode, but information from completed episodes is neither stored nor reused in subsequent tasks.
- **Voyager:** Voyager (Wang et al., 2024a) maintains a growing library of reusable skills learned from interaction experience. Relevant skills are retrieved according to the current task and provided to the agent as prior experience for subsequent decision making. We adapt its skill-memory interface to our benchmarks while preserving this continual skill accumulation and retrieval mechanism.
- **MemoryBank:** MemoryBank (Zhong et al., 2024) maintains long-term memory through explicit reinforcement and forgetting. Its memory update mechanism is inspired by the Ebbinghaus forgetting curve, allowing stored information to be strengthened or discarded according to its importance and temporal decay. Relevant retained memories are retrieved as additional context for later tasks.

- **Generative Agents:** Generative Agents (Park et al., 2023) stores past observations in a memory stream and retrieves them according to relevance, recency, and importance. It further synthesizes higher-level reflections from accumulated experience, which provide abstract context for subsequent decisions. We retain both its retrieval and reflection mechanisms under our common task interface.
- **Agent Workflow Memory:** Agent Workflow Memory (AWM) (Wang et al., 2025) extracts reusable workflows from previous task trajectories and stores them as procedural knowledge. For a new task, relevant workflows are retrieved and provided to the agent as guidance for action generation. We use its online setting, where workflows are constructed only from episodes completed earlier in the same run.
- **ExpeL:** ExpeL (Zhao et al., 2024) collects successful and failed trajectories from training tasks and distills them into general natural-language insights. During inference, the extracted insights are provided to the agent, together with similar successful trajectories retrieved according to task similarity as few-shot examples. Following the original protocol, we use four-fold validation in all experiments.

A.3 Implementation Details

All methods share the same agent interface and benchmark evaluation settings, while retaining their original memory construction and update mechanisms. For HieraMem, each run starts from empty memory and only completed episodes can influence subsequent episodes. Model parameters remain frozen throughout evaluation. The Qwen backbones use `Q8_0` quantization and run on NVIDIA GeForce RTX 2080 Ti GPUs. All LLM calls use temperature 0.

The maximum number of interaction steps is 30 for ALFWorld, ScienceWorld, and PDDL, 8 for HotpotQA, and 12 for FEVER. The maximum output length for agent action generation is 128 tokens for ALFWorld, 512 tokens for ScienceWorld and HotpotQA, and 160 tokens for FEVER and PDDL. All other LLM calls use a maximum output length of 4,096 tokens across all benchmarks. For HieraMem, we set $K_{\mathrm{e}} = 3$ for memory evolution and use $w = 3$, $K_{\mathrm{mem}} = 1$, and $\delta = 0.6$ as the default settings for decision-level retrieval. Sensitivity analyses for the decision-level retrieval parameters are provided in Section B.2.

B Additional Analyses

B.1 Fine-Grained Analysis on ALFWorld

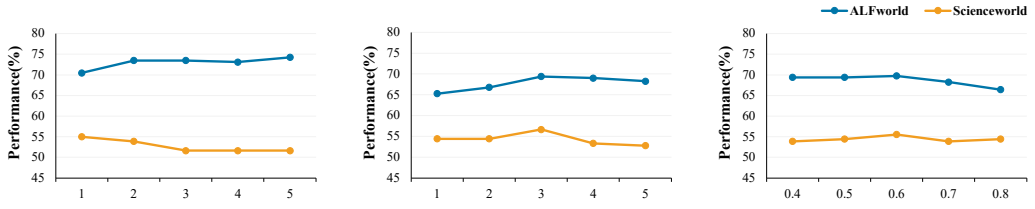
Table 4: Success rate (SR, %) across ALFWorld task types with Qwen3-14B. Overall performance additionally reports average steps.

Method	Overall SR	Avg. Steps	Put	Clean	Heat	Cool	Examine	PutTwo
No-memory	57.46	19.99	79.17	74.19	47.83	66.67	50.00	5.88
Voyager	66.42	16.45	87.50	70.97	47.83	95.24	66.67	17.65
MemoryBank	56.72	18.61	83.33	54.84	47.83	85.71	50.00	5.88
Generative	65.67	17.11	91.67	70.97	60.87	76.19	55.56	23.53
AWM	60.70	18.75	79.17	63.44	46.38	80.95	77.78	5.88
ExpeL	65.92	16.51	68.06	69.89	68.12	79.37	68.52	33.33
HieraMem	70.90	17.62	87.50	69.89	66.67	80.95	62.96	**50.98**

We further analyze performance across the six ALFWorld task types under the Qwen3-14B backbone. As shown in Table 4, HieraMem improves the overall success rate from 57.46 to 70.90 over No-memory, an absolute gain of 13.44 points. Meanwhile, the average number of interaction steps decreases from 19.99 to 17.62. Thus, the higher success rate is not obtained by simply extending the interaction trajectory. At the task-type level, HieraMem improves five of the six categories over No-memory. The largest gain occurs on *PutTwo*, where the success rate increases from 5.88 to 50.98 (+45.10). Substantial improvements are also observed on *Heat* (+18.84), *Cool* (+14.28), *Examine* (+12.96), and *Put* (+8.33), while *Clean* decreases by 4.30 points.

The task-level breakdown highlights the strength of HIERAMEM on more complex ALFWorld tasks. Different methods achieve their strongest results on different categories. For example, Generative performs better on *Put*, ExpeL on *Heat*, Voyager on *Cool*, and AWM on *Examine*. In comparison, HIERAMEM achieves the best performance on *PutTwo*, reaching 50.98 compared with 33.33 for the next-best baseline, while remaining competitive on *Clean*, *Heat*, and *Cool*. These tasks involve multi-stage or coordinated interactions, suggesting that the evidence-anchored memory of HIERAMEM is particularly beneficial when successful execution requires maintaining and reusing useful experience across multiple decisions.

B.2 SENSITIVITY ANALYSIS



(a) K_{mem}. (b) w. (c) δ.

Figure 5: Parameter sensitivity analysis of HIERAMEM.

We study the sensitivity of HIERAMEM to the retrieval budget K_{mem}, local history window w, and similarity threshold δ, as shown in Figure 5. The two benchmarks favor different retrieval budgets. ALFWorld performs better with a slightly larger retrieval budget, improving from $K_{\text{mem}} = 1$ to $K_{\text{mem}} = 2$ and remaining relatively stable thereafter, whereas ScienceWorld performs best at $K_{\text{mem}} = 1$ and declines before plateauing at $K_{\text{mem}} \geq 3$. For the local history window, both benchmarks achieve their best observed performance at $w = 3$ and decline with longer windows, suggesting that additional history beyond this point offers no further benefit in the tested settings. For the similarity threshold, both benchmarks peak at $\delta = 0.6$. Performance varies modestly between $\delta = 0.4$ and 0.6, after which ALFWorld declines while ScienceWorld shows smaller, non-monotonic variations. Overall, we use the shared configuration $K_{\text{mem}} = 1$, $w = 3$, and $\delta = 0.6$ throughout the experiments without benchmark-specific adjustment. The contrasting responses to retrieval breadth suggest that different environments favor different retrieval budgets, highlighting the potential for further gains through environment-specific tuning or adaptive budget selection.

B.3 TOKEN COST ANALYSIS

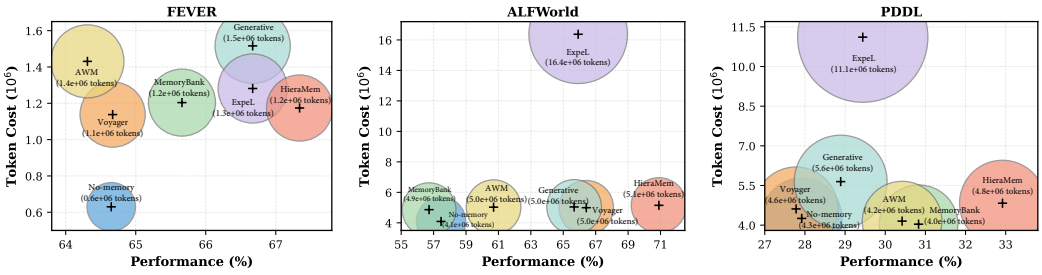


Figure 6: Performance versus token cost across different memory methods.

We further report the performance–token cost trade-off on FEVER, ALFWorld, and PDDL in Figure 6. The results extend the observations on ScienceWorld and HotpotQA reported in Section 5.5. Across all three benchmarks, HIERAMEM achieves the highest performance without incurring substantially higher token consumption than most memory-based baselines. Although memory formation and evolution introduce additional LLM calls, memory utilization relies on compact Evidence Anchors and their associated Reusable Memory rather than repeatedly injecting full trajectories, thereby keeping the runtime context compact. ExpeL incurs notably higher token costs on ALFWorld and PDDL, mainly due to its reflection-intensive experience collection stage, where failed trajectories trigger additional reflection calls.

B.4 VISUALIZATION OF MEMORY EVOLUTION

To provide a concrete view of how memory evolves in HIERAMEM, Figure 7 visualizes the memory hierarchy and representative evolution operations on ALFWorld. The complete hierarchy in Figure 7a shows the accumulated associations among *Episodic Evidence*, *Evidence Anchors*, and *Reusable Memory*. For the three illustrative evolution examples, we omit the outer *Episodic Evidence* layer and show only the anchor–memory structures involved in each operation. In Figure 7b, ADD retains the candidate as an independent reusable memory because comparison with the failure anchors of three semantically related neighbors reveals a distinct local correction. In Figure 7c, REWRITE consolidates the candidate with neighboring memories that address the same local correction under compatible decision conditions. The gray triangle marks a memory selected for REWRITE. After REWRITE, it no longer participates in subsequent retrieval, while its supporting failure anchor associations are inherited by the revised memory. In Figure 7d, NOOP leaves the persistent hierarchy unchanged: the candidate and its newly identified anchors are used for the evolution decision but are not incorporated. Together, these operations illustrate how evidence-aware evolution allows HIERAMEM to preserve distinct corrections, consolidate compatible knowledge, and avoid redundant accumulation while retaining evidence associations across the memory hierarchy.



(a) Evolved memory hierarchy.

(b) ADD.

(c) REWRITE.

(d) NOOP.

Figure 7: Visualization of the evolved memory hierarchy and representative cross-episode evolution decisions on ALFWorld. (a) The complete hierarchy after continual interaction. (b)–(d) Representative ADD, REWRITE, and NOOP outcomes. Highlighted nodes and edges illustrate the memory and evidence associations involved in each evolution operation.

C PSEUDOCODE

Algorithm 1: HIERAMEM: Online Memory Formation, Evolution, and Utilization

Input: Task sequence $\{\mathcal{T}_i\}_{i=1}^{N}$ with goals $\{g_i\}$; policy π_θ; w, K_{mem}, δ; L_i denotes the maximum step budget for task $\mathcal{T}_i$.

Output: Memory hierarchy $\mathcal{H}_N = (\mathcal{E}_N, \mathcal{A}_N, \mathcal{M}_N; \mathcal{R}_N^{EA}, \mathcal{R}_N^{AM})$

$\mathcal{H}_0 \leftarrow (\varnothing, \varnothing, \varnothing; \varnothing, \varnothing)$ *// Initialize the memory hierarchy*

for $i \leftarrow 1$ **to** N **do**

  $\mathcal{H}_i \leftarrow \mathcal{H}_{i-1}$ *// Inherit the previous memory state*

  // Dual-Granularity Memory Utilization

  $A_i^{+*} \leftarrow \arg\max_{A^+ \in \mathcal{A}_{i-1}^+} \text{sim}(g_i, g(A^+))$ *// Select a task-level global reference*

  for $t \leftarrow 1$ **to** L_i **do**

    $q_i^{(t)} \leftarrow \text{Serialize}(g_i, h_i^{(w,t)})$ *// Represent the current local context*

    $\mathcal{A}_i^{-*(t)} \leftarrow \text{TopK}_{K_{\text{mem}}}\{A^- \in \mathcal{A}_{i-1}^- : \text{sim}(q_i^{(t)}, A^-) \geq \delta\}$ *// Retrieve relevant failure anchors*

    $\mathcal{M}_i^{*(t)} \leftarrow \{m : \exists A^- \in \mathcal{A}_i^{-*(t)}, (A^-, m) \in \mathcal{R}_{i-1}^{AM}\}$ *// Access anchor-linked reusable memory*

    $c_i^{(t)} \leftarrow \text{Context}(A_i^{+*}, \mathcal{M}_i^{*(t)})$ *// Combine global and local memory*

    $a_i^{(t)} \sim \pi_\theta(\cdot \mid g_i, h_i^{(t)}, c_i^{(t)})$ *// Select the next action*

    Execute $a_i^{(t)}$ and update $h_i^{(t)}$ *// Advance the interaction*

  // Evidence-Anchored Memory Formation

  $E_i \leftarrow (g_i, \tau_i, y_i)$ *// Construct the completed episode*

  $\mathcal{E}_i \leftarrow \mathcal{E}_{i-1} \cup \{E_i\}$ *// Archive episodic evidence*

  if $y_i = 0$ **then**

    if $\mathcal{E}_{i-1}^+ \neq \varnothing$ **then**

      $\widehat{E}_i^+ \leftarrow \arg\max_{E \in \mathcal{E}_{i-1}^+} \text{sim}(g_i, g(E))$ *// Find a related successful episode*

      $(A_i^-, A_i^+, \widetilde{m}_i) \leftarrow \mathcal{C}_{\text{LLM}}^{\text{form}}(E_i, \widehat{E}_i^+; \mathcal{P}_f)$ *// Form evidence anchors and candidate memory*

    else

      $\widehat{E}_i^+, A_i^+ \leftarrow \varnothing$ *// No successful reference or anchor*

      $(A_i^-, \widetilde{m}_i) \leftarrow \mathcal{C}_{\text{LLM}}^{\text{form}}(E_i; \mathcal{P}'_f)$ *// Form from failure evidence alone*

    if $\widetilde{m}_i \neq \varnothing$ **then**

      // Cross-Episode Memory Evolution

      $\mathcal{N}_i \leftarrow \text{TopK}_e(\mathcal{M}_{i-1}; m \mapsto \text{sim}(z_i, z(m)))$ *// Retrieve related existing memories*

      $(d_i, \mathcal{S}_i, \widehat{m}_i) \leftarrow \mathcal{C}_{\text{LLM}}^{\text{evolve}}\big((\widetilde{m}_i, A_i^-), \{(m, \mathcal{A}_{i-1}^-(m)) : m \in \mathcal{N}_i\}; \mathcal{P}_e\big)$ *// Select operation*

      $\mathcal{H}_i \leftarrow \text{Update}(\mathcal{H}_i, \widehat{E}_i^+, A_i^-, A_i^+, \widetilde{m}_i, d_i, \mathcal{S}_i, \widehat{m}_i)$ *// Update memory and evidence associations*

return $\mathcal{H}_N$ *// Return the accumulated hierarchy*

D PROMPTS

D.1 MEMORY FORMATION

The following prompt identifies *Evidence Anchors* and derives a candidate *Reusable Memory* from a contrastive episode pair. Here, `[[benchmark_specific_rules]]` provides the benchmark-specific action space, operational constraints, and semantics needed to interpret observations and state transitions.

Extraction Prompt

```
You are an agent-memory analyst extracting ONE contrastive memory pattern. Memory is a reusable local
      corrective rule that helps a future agent choose a better next control move. It is not an episode
      summary, task recipe, route, or full plan.

You are given one failed trajectory and one successful trajectory from a similar task.

# Contrastive extraction objective:
```

```
## Compare the successful and failed trajectories to infer the reusable local decision principle that
      explains their behavioral difference, and express it as one concise runtime guidance.
## Prefer the earlier control error over later repeated loops if the later loop follows from that error.

# Abstraction rules:
## MECH: abstract reusable mechanism, not task-specific.
## ACT: abstract local decision stages being controlled, not a raw action sequence or route.
## SYM: 2-5 observable abstract trigger cues plus the relevant control family.
## CAUSE: explain the decisive state difference between the failed and successful branches.
## GUIDANCE_TEXT: give a concise state-aware branch replacement rule: when the trigger appears, avoid the
      failed branch and take the corrective branch after preserving or establishing the needed state.
## PRIMARY_ANCHOR_KEY_STEP must be from the failed trajectory and directly prove the mechanism.

# Requirements:
## Do not write guidance that only says check, verify, confirm, inspect, reconsider, diagnose, retry, or try
       another strategy.
## DO NOT use concrete object names, product IDs, exact locations, exact page text, exact counts, and
      episode-specific commands unless essential.
## Do not infer hidden absence, invalidity, prerequisites, global unavailability, completion, or required
      command form.
## Do not claim success progress unless the observation shows state change, successful interaction, or task-
      relevant advancement.
## GUIDANCE_TEXT must not recommend a corrective branch if the same action family under the same relevant
      state/context signature already appears later in the failed trajectory and still yields no progress,
      invalid feedback, unchanged observation, unchanged result set, unchanged page state, or no task-
      relevant state change.
## If the same action family can become correct only after a changed precondition or context, the guidance
      must explicitly state that required precondition/context change before recommending the action.
## Prefer the decisive state difference over surface action wording: identify what state the failed branch
      destroys/ignores and what state the successful branch preserves/establishes.
## Prefer state contrast over action wording: the corrective branch must explain the decisive state that
      made the successful action work, not only rewrite the failed action into a different surface form.
## Do not treat a state as corrective if the failed trajectory already reached the same state and still
      failed without task-relevant progress.

# Successful-evidence requirements:
## If a successful trajectory is provided, extract SUCCESS_EVIDENCE_STEPS that show the clearest task-
      relevant state changes.
## SUCCESS_EVIDENCE_STEPS must prioritize action-outcome steps that produce an observation-confirmed task-
      relevant state transition.
## SUCCESS_EVIDENCE_STEPS do not need to exhibit the same trigger, cause, or recovery branch as
      PRIMARY_ANCHOR_KEY_STEP.
## Prefer state-changing steps over navigation-only, inspection-only, or thought-only steps.
## SUCCESS_EVIDENCE_TEXT must summarize the abstract task progress supported by the selected steps.
[[benchmark_specific_rules]]

# Content for analysis:

# Failed Episode:
TASK_GOAL:
[[failed_goal]]

TRAJECTORY:
[[failed_trajectory]]

# Successful Reference Episode:
TASK_GOAL:
[[success_goal]]

TRAJECTORY:
[[success_trajectory]]

# Output format
Return exactly:

PATTERN
MECH: UPPER_SNAKE_CASE reusable mechanism name
ACT: comma-separated abstract decision/action stages
SYM: 2-5 abstract runtime trigger cues separated by |
CAUSE: one sentence explaining the contrastive reusable mechanism
GUIDANCE_TEXT: one concise runtime guidance sentence
PRIMARY_ANCHOR_KEY_STEP: decisive evidence-bearing step t from the failed trajectory
```

```
SUCCESS_EVIDENCE_STEPS: successful trajectory step ids separated by |, or null
SUCCESS_EVIDENCE_TEXT: one concise abstract summary of task progress supported by those steps, or null

If no valid reusable contrastive pattern is supported, return exactly:
NO_PATTERN
```

D.2 MEMORY EVOLUTION

The following prompt compares a candidate *Reusable Memory* with related existing memories and their supporting failure anchors to determine the appropriate cross-episode evolution decision.

Memory Evolution Prompt

```
You are a contrastive memory evolution curator. Compare one candidate pattern with retrieved existing
      memories, and choose exactly one evolve operation: ADD, REWRITE, or NOOP.

# Core criterion:
Two patterns are the same memory only when they describe the same reusable local correction across MECH, ACT,
       SYM, CAUSE, and GUIDANCE_TEXT. CAUSE explains the correction mechanism, but it must be consistent with
       the decision stage in ACT, the trigger boundary in SYM, and the next control move in GUIDANCE_TEXT.

# Compare:
## MECH: reusable correction mechanism;
## ACT: local decision stage being controlled;
## SYM: runtime trigger and evidence boundary;
## CAUSE: why the failed branch is wrong and why the corrective branch helps;
## GUIDANCE_TEXT: what behavior/control branch to stop or preserve, and what next abstract control branch to
       take.

# Choose exactly one operation:
1. ADD: Use ADD when the candidate would change future retrieval or guidance by adding a materially
      different trigger, decision stage, failed branch, next control branch, or correction mechanism. Use ADD
       if CAUSE is similar in topic but ACT, SYM, or GUIDANCE_TEXT point to a different local control problem
      . Use ADD only if the candidate is evidence-supported and directly usable as an independent memory.
2. REWRITE: Use REWRITE only when the candidate and one or more neighbors express the same local correction:
       same trigger boundary, same decision stage, same failed branch, same next control branch, and same
      mechanism. When rewriting, use CAUSE to unify the correction mechanism, use GUIDANCE_TEXT to unify the
      runtime control move, and keep the rewritten pattern consistent with ACT and SYM.
3. NOOP: Use NOOP when the candidate is already covered, weaker, vaguer, unsupported, too episode-specific,
      route-like, only changes wording, or only describes successful trajectory facts without changing the
      future local control decision.

# Requirements:
## Do not REWRITE just because MECH names, objects, no-progress feedback, or broad CAUSE topics are similar.
## Do not relax hard task constraints, change required target/entity type, infer hidden absence, or assume
      global unavailability.
## Generalize concrete objects, products, locations, UI text, and episode-specific commands into reusable
      roles.
## REWRITE_SOURCE_IDS must contain only NODE_ID values explicitly listed in Retrieved Neighbor Patterns.

# Candidate Pattern:
NODE_ID: candidate
FAILURE_ANCHOR:
[[candidate_failure_anchor]]
MECH: [[candidate_mech]]
ACT: [[candidate_act]]
SYM: [[candidate_sym]]
CAUSE: [[candidate_cause]]
GUIDANCE_TEXT: [[candidate_guidance]]

# Retrieved Neighbor Patterns:
[[for each retrieved neighbor]]
NODE_ID: [[existing_pattern_id]]
FAILURE_ANCHORS:
[[supporting_failure_anchors]]
MECH: [[existing_mech]]
ACT: [[existing_act]]
SYM: [[existing_sym]]
CAUSE: [[existing_cause]]
GUIDANCE_TEXT: [[existing_guidance]]
```

```
# Output:
OPERATION: ADD | REWRITE | NOOP
REWRITE_SOURCE_IDS: [] or existing_pattern_id_1|existing_pattern_id_2|...
PRIMARY_ANCHOR_SOURCE: candidate or selected existing pattern id for REWRITE, otherwise null
MECH: UPPER_SNAKE_CASE reusable mechanism name or null
ACT: comma-separated abstract decision/action stages or null
SYM: 2-5 abstract runtime trigger cues separated by | or null
CAUSE: one sentence explaining the reusable local correction mechanism or null
GUIDANCE_TEXT: one concise runtime guidance sentence or null

Return exactly in this format and nothing else.

If OPERATION is ADD, REWRITE_SOURCE_IDS must be [] and all pattern fields must be null.
If OPERATION is NOOP, REWRITE_SOURCE_IDS must be [] and all pattern fields must be null.
If OPERATION is REWRITE, REWRITE_SOURCE_IDS must contain one or more existing pattern IDs separated only by
      |, PRIMARY_ANCHOR_SOURCE must be candidate or one of those IDs, and all pattern fields must contain the
       rewritten merged pattern.
```