EVOSPACE: CO-EVOLVING THE HARNESS AND EDIT SPACE FOR MEMORY-GROUNDED QUESTION ANSWERING

Anonymous authors
Paper under double-blind review

ABSTRACT

The performance of LLM-based agents depends not only on their underlying models but also on the harnesses that orchestrate their interactions with the environment. Recently, researchers have focused on harness self-evolution, in which agents iteratively refine their harnesses using operational experience within a harness edit space. However, dynamically aligning this edit space with evolving adaptation needs remains a central challenge. A fixed harness edit space may preclude necessary updates, whereas allowing all permissible edits initially increases the complexity of selecting and combining updates. We introduce EvoSpace to address this challenge by jointly evolving the agent harness and its corresponding edit space. As the harness evolves, feedback-driven diagnosis guides EvoSpace to progressively expand the permissible updates across components, scopes, and operations. This enables changes beyond the initial space without exposing all combinations from the outset. Across two established benchmarks, EvoSpace improves over the compared memory systems. On LoCoMo, EvoSpace achieves average relative gains across F1 and BLEU-1 of 8.62% with qwen3.6-flash and 13.71% with GPT-5.1 over EvolveMem. From the experimental analysis, our key insight is that the harness edit space governs which adaptations are possible, making its dynamic alignment with evolving adaptation needs essential for effective self-evolution.

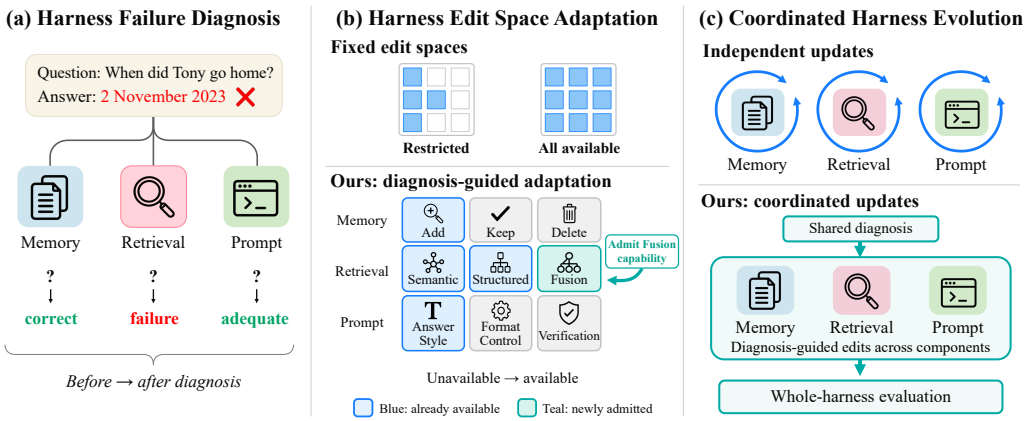


Figure 1: **EvoSpace: Harness evolution through an adaptive edit space.** (a) Failure diagnosis identifies likely component-level causes of incorrect answers. (b) Diagnosis guides edit-space expansion by admitting relevant, previously unavailable editing capabilities. (c) Shared diagnostic evidence and whole-harness evaluation coordinate successive updates across Memory, Retrieval, and Prompt.

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.

1 INTRODUCTION

Large language models (LLMs) are increasingly deployed as agents to perform complex tasks through reasoning, tool use, and interaction with external environments (Yao et al., 2023; Wang et al., 2024; Wu et al., 2024b). Their performance depends not only on the underlying models but also on the *agent harness*, the surrounding software system that governs how a model operates in the real-world environment through prompts, tools, memory mechanisms, and execution logic (Zhang et al., 2026b; Xia et al., 2026; Lee et al., 2026). *Harness self-evolution* iteratively improves this system using execution feedback while keeping the underlying model parameters fixed (Zhang et al., 2026b; Xia et al., 2026).

Harness self-evolution operates within a *harness edit space*, the set of implementation-supported modifications currently permitted by the available editing operations and their scopes. Existing studies improve harnesses through agent implementation search (Hu et al., 2025a; Zhang et al., 2026d; 2024; Lee et al., 2026), trace-guided repair (Zhang et al., 2026b; Chen et al., 2026; Cai et al., 2026), and memory adaptation (Zhang et al., 2026a; Liu et al., 2026b; Zhou et al., 2026; Zhang et al., 2026c; Lin et al., 2026), while other approaches regulate edit proposals, candidate selection, and editing permissions (Xia et al., 2026; Zhao et al., 2026). However, an unsuccessful task outcome alone may not reveal which component requires modification, and an update to one component can alter what is needed from the others (Cheng et al., 2024; Yuksekgonul et al., 2024). A fixed, restricted edit space may therefore exclude changes required to address diagnosed failures, whereas making all supported edits available from the outset can make effective updates harder to identify under a limited evolution budget. Moreover, improvements to individual components do not necessarily translate into better overall task performance (Opsahl-Ong et al., 2024; Cheng et al., 2024). The central challenge is thus twofold: aligning the available edit space with evolving needs identified through diagnosis, and assessing successive updates by their effects on the complete harness.

We introduce **EvoSpace** to address this challenge through diagnosis-guided adaptation of the harness edit space before each harness update is selected. We instantiate EvoSpace for long-term memory-grounded question answering, focusing on three editable harness components: Memory, Retrieval, and Prompt, which govern information retention, evidence selection, and answer generation, respectively. Figure 1 highlights three complementary aspects of EvoSpace. *Harness Failure Diagnosis* (Figure 1(a)) uses answer traces and quality scores to identify likely component-level failures and the evidence supporting them. Guided by this diagnosis, *Harness Edit Space Adaptation* (Figure 1(b)) determines whether the diagnosis supports a relevant, preimplemented capability that is not yet available in the current edit space. If so, it prioritizes admitting that capability over further tuning existing capabilities; otherwise, it retains the current space. Admitting a capability expands the available edit space without itself modifying the harness. *Coordinated Harness Evolution* (Figure 1(c)) uses shared diagnostic evidence to guide edits that may span multiple components and evaluates the resulting candidate harness as a whole. This feedback loop connects edit-space adaptation and edit selection through shared diagnostic evidence. Whole-harness evaluation coordinates successive updates across Memory, Retrieval, and Prompt according to their effects on overall task performance.

We evaluate EvoSpace on LoCoMo (Maharana et al., 2024) and MemBench (Tan et al., 2025a) across multiple task domains using two LLM backbones. On LoCoMo, EvoSpace improves over EvolveMem by an average of 8.62% and 13.71% in relative terms across F1 and BLEU-1 with qwen3.6-flash and GPT-5.1, respectively. On LoCoMo, adaptive edit-space expansion also outperforms both fixed and fully available spaces under the same evolution budget. From the experimental analysis, our key insight is that the harness edit space governs which adaptations are possible, making its dynamic alignment with evolving adaptation needs a key factor in effective self-evolution.

Our contributions are summarized as follows:

- **Harness Edit-space Evolution.** We make the available modifications an evolving state rather than a fixed design choice through diagnosis-guided harness edit-space evolution. This allows the edit space to adapt to the needs revealed during harness self-evolution.

- **Coordinated Harness Evolution.** EvoSpace connects component-level failure diagnosis, edit-space adaptation, and edit selection through shared diagnostic evidence. Whole-harness evaluation coordinates successive updates across Memory, Retrieval, and Prompt.
- **Experimental Evaluation and Insights.** We evaluate EvoSpace on two benchmarks with two LLM backbones, achieving average relative improvements across LoCoMo F1 and BLEU-1 of 8.62% and 13.71% over EvolveMem with qwen3.6-flash and GPT-5.1, respectively. Further analysis on LoCoMo demonstrates the benefit of adaptive expansion over fixed and fully available edit spaces under the same evolution budget, highlighting the importance of dynamically adapting the available modifications rather than simply maximizing their range.

2 Related Work

Long-term Conversational Memory. Agent memory retains information for later use, with frameworks and surveys characterizing its components, functions, and lifecycle (Sumers et al., 2024; Zhang et al., 2025; Hu et al., 2025b). Conversational memory supports recall across sessions (Xu et al., 2022; Wang et al., 2023; Wu et al., 2024a). Memory organization ranges from reflections over recorded experiences to conversation segments and coordinated memory types (Park et al., 2023; Pan et al., 2025; Wang & Chen, 2025). Systems implement tiered storage, consolidation, associative links, and semantic compression (Packer et al., 2023; Chhikara et al., 2025; Xu et al., 2025; Liu et al., 2026a). Maintenance mechanisms include forgetting, belief updates, and reflective refinement of conversational retrieval (Zhong et al., 2024; Kynoch et al., 2023; Tan et al., 2025b). Related long-context reading methods organize text into hierarchical summaries or gist memories and selectively revisit source content (Chen et al., 2023; Lee et al., 2024). Beyond retrieving external knowledge for generation (Lewis et al., 2020), adaptive methods condition retrieval on generation state (Jiang et al., 2023; Asai et al., 2024), evidence quality (Yan et al., 2024), or query complexity (Jeong et al., 2024). Our Harness builds on these mechanisms for its Memory and Retrieval components.

Agent Harness Evolution. An agent harness is the runtime infrastructure around a language model, including context management, tool interfaces, and execution logic (Lee et al., 2026; Zhang et al., 2026b; Chen et al., 2026). Related optimization methods target its individual components. Prompt optimizers use evaluated candidates or natural-language feedback to revise instructions (Pryzant et al., 2023; Yang et al., 2024; Agrawal et al., 2026); other methods iteratively refine model outputs or optimize text-valued components using feedback (Madaan et al., 2023; Yuksekgonul et al., 2024; Lee et al., 2025). Multi-stage program optimizers use downstream objectives and, in Trace, execution traces to guide updates (Opsahl-Ong et al., 2024; Cheng et al., 2024). Program search uses language models to propose executable variants and retains candidates based on evaluation (Lehman et al., 2023; Novikov et al., 2025), while agent-design and workflow search optimize broader agentic systems (Hu et al., 2025a; Zhang et al., 2024). Context adaptation accumulates reusable strategies in evolving playbooks or memory (Zhang et al., 2026e; Suzgun et al., 2025); Meta Context Engineering further co-evolves context-engineering skills and context artifacts (Ye et al., 2026). At the harness level, Darwin Gödel Machine (Zhang et al., 2026d) modifies agent code and maintains evaluated candidates, while Live-SWE-agent (Xia et al., 2025) evolves custom tools during execution. Self-Harness, HarnessFix, and MOSS use execution traces or production failures to guide implementation changes and validate candidate repairs (Zhang et al., 2026b; Chen et al., 2026; Cai et al., 2026). Meta-Harness (Lee et al., 2026) uses prior code, scores, and traces to search harness implementations spanning prompts, retrieval, and state updates. RobustSGPO (Zhao et al., 2026) additionally controls editing permissions, scopes, and operations during harness search. These methods optimize runtime behavior and aspects of the improvement process. EvoSpace makes the available edit space explicit through a representation of edits by component, task scope, and operation. A shared diagnosis guides both space expansion and subsequent edit selection, with whole-Harness evaluation determining retention or rollback.

Self-Evolving Memory. Experience-based methods turn execution feedback into reusable reflections or insights (Shinn et al., 2023; Zhao et al., 2024). Memory-R1 (Yan et al., 2026) learns memory-management and answer policies; Agentic Memory (Yu et al., 2026) unifies long- and short-term memory operations. MemSkill (Zhang et al., 2026c) evolves reusable memory skills, and MemMA (Lin et al., 2026) converts probe failures into local memory repairs.

MemEvolve (Zhang et al., 2026a) jointly evolves experiential knowledge and memory architectures. EvolveMem (Liu et al., 2026b) uses per-question failure logs to change retrieval and answer-generation configurations, introduce configuration dimensions, and trigger memory re-extraction. EvoSpace represents the runtime Harness and its available edit space separately. Diagnosis first determines whether to retain or expand that space, then guides update selection across Memory, Retrieval, and Prompt.

3 METHODOLOGY

EvoSpace jointly evolves a runtime Harness and its *harness edit space* for long-term memory-grounded question answering, while keeping the answer model fixed. The Harness governs the agent's behavior; the edit space specifies the capabilities available to modify it. EvoSpace connects capability admission and concrete operation generation through a shared diagnosis, which guides both decisions: which editing capabilities to admit and how to instantiate available capabilities into concrete operations. Candidate evaluation determines the retained Harness and feedback for the next round. Figure 2 shows the architecture, and Algorithm 1 specifies the complete procedure. Appendix E gives condensed prompt templates for extraction, answering, and evolution.

3.1 PROBLEM FORMULATION AND OVERVIEW

Let $\mathcal{D} = \{(x_i, y_i^\star)\}_{i=1}^N$ be a task set, where $x_i = (h_i, q_i)$ contains an interaction history and a query, and $y_i^\star$ is the reference answer. The Harness $H = (M, R, P)$ comprises Memory, Retrieval, and Prompt, which govern information retention, evidence selection, and instructions for evidence use, answer format, and verification, respectively. Harness evolution optimizes H while fixing the answer-model parameters θ (Xia et al., 2026). Let $\hat{y}_i = \mathrm{LLM}_\theta(H, x_i)$ denote its answer. We measure empirical quality by

$$\mathcal{J}(H; \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} \mathrm{SCORE}(\hat{y}_i, y_i^\star)\,. \tag{1}$$

Let $\mathcal{O}$ be a fixed library of preimplemented editing capabilities, and $\mathcal{E}_t \subseteq \mathcal{O}$ the capabilities available at round t. A capability e specifies a target component $\chi \in \{M, R, P\}$, an operation type o, and an allowed argument domain $\mathcal{A}_e$; a concrete operation a assigns arguments to that capability:

$$e = (\chi, o, \mathcal{A}_e), \qquad a = (e, \alpha), \qquad \alpha \in \mathcal{A}_e. \tag{2}$$

Capabilities may permit specific configuration operations, such as setting RRF fusion. Admitting one expands $\mathcal{E}_t$; executing it changes H_t. Changing arguments within an available capability does not expand the space.

We initialize H_0 with memories extracted from the histories, BM25 retrieval, and a minimal base prompt. The initial space $\mathcal{E}_0$ matches EvolveMem's available editing capabilities, including Retrieval configuration and Prompt updates. Capability availability is separate from the BM25-only runtime initialization. Given a round budget T, per-round operation limit n, and rollback tolerance $\tau_{\mathrm{rev}} \geq 0$, EvoSpace returns the best observed retained Harness H^{best}.

3.2 HARNESS-GROUNDED ANSWERING AND DIAGNOSIS

Given x_i, the agent retrieves evidence $E_i = \mathrm{Retrieve}(q_i; M, R)$ and answers with $\mathrm{LLM}_\theta(q_i, E_i; P)$. EVALUATE runs this pipeline on $\mathcal{D}$, returning $F = \mathcal{J}(H; \mathcal{D})$ and diagnostic input $\mathcal{B}$. The latter combines correctness feedback with traces recording retrieved evidence and provenance, Prompt configuration, verification decisions, and answers, prioritizing completely incorrect samples. Reference answers are used for scoring but excluded from diagnosis inputs.

The diagnosis agent examines $(\mathcal{B}, H)$ and produces a failure analysis supported by trace evidence, one prioritized component $\chi \in \{M, R, P\}$, and confidence in that attribution. Within the shared diagnosis, component localization prioritizes capability admission, while failure analysis guides concrete operations that may jointly modify different components. Execution arguments are determined only during editing.

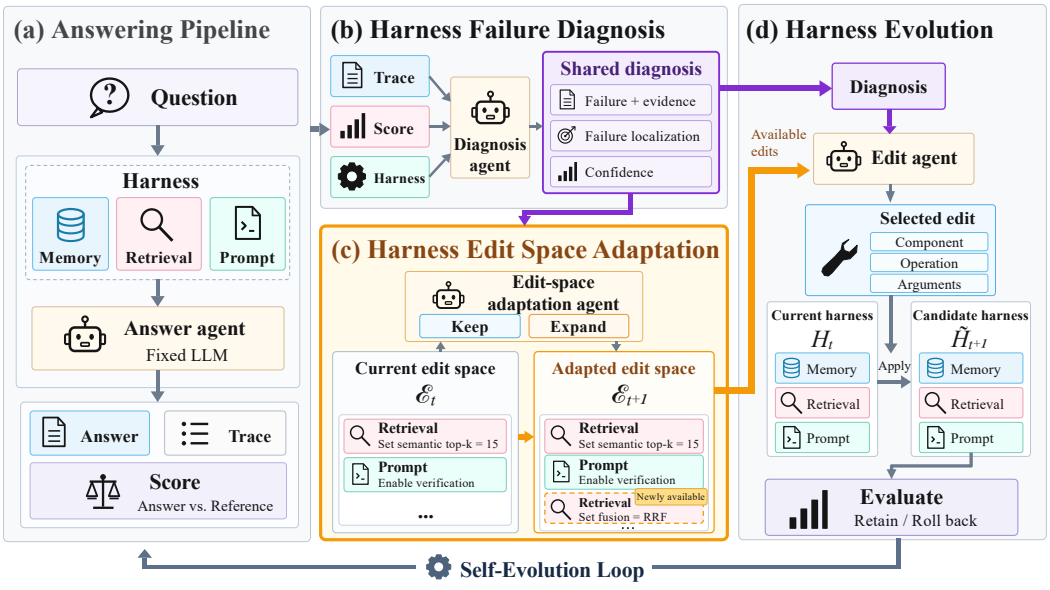


Figure 2: **EvoSpace framework for joint Harness and edit-space evolution.** (a) A fixed answer model generates responses using the Memory, Retrieval, and Prompt components of the current Harness. (b) The diagnosis agent analyzes answer scores and execution traces to identify potential component-level failures. (c) The shared diagnosis guides retention or expansion of the edit space. (d) The edit agent instantiates available capabilities into concrete operations, producing a candidate Harness that is evaluated for retention or rollback. Evaluation feedback informs subsequent evolution rounds, while newly admitted capabilities persist across rollbacks.

3.3 Harness Edit-space Evolution

The meta-evolution agent uses the shared diagnosis to *keep* or *expand* the available edit space. When the diagnosis supports a relevant unavailable capability, it prioritizes admission over further tuning of existing capabilities; otherwise, it keeps the space unchanged. For example, missing evidence may motivate admitting Memory repair while existing Retrieval and Prompt capabilities remain available.

Expansion admits at most one capability from $\mathcal{O} \setminus \mathcal{E}$ for the prioritized component χ. This expands the operations available to the editor, without modifying the current Harness or assigning execution arguments. The admitted capability is also passed to EDIT as a priority hint ($e^{\text{new}} = \perp$ on keep), not a mandatory operation. Memory and retrieval formulations, edit semantics, and the initial interface are detailed in Appendix A. Appendix B.1 describes these changes on LoCoMo with qwen3.6-flash: the initial capability set is retained through R4, Memory repair is admitted at R5, and the expanded set is retained through R6–R7.

3.4 Diagnosis-guided Harness Evolution

Concrete operations. Using the diagnosis, updated edit space, and admission priority, EDIT generates an ordered sequence of at most n operations per round. These operations form a single candidate Harness and may target different components. Each operation instantiates an available capability with concrete arguments. A component-level configuration edit may update multiple parameters or category-specific settings together and counts as one operation. For Memory repair, $\mathcal{S}_t$ supplies the target questions' complete conversation histories to ground the memory content. A batch repair counts as one operation even if it covers multiple questions and adds multiple memories. The controller checks capability membership, argument constraints, and required source references. Empty or invalid sequences leave the Harness unchanged.

Algorithm 1 EvoSpace: Joint Harness and Edit-space Evolution

Require: $\mathcal{D}, \theta, H_0, \mathcal{E}_0, \mathcal{O}, T = 7, n = 2, \tau_{\text{rev}}$
$H \leftarrow H_0$; $\mathcal{E} \leftarrow \mathcal{E}_0$
$(F, \mathcal{B}) \leftarrow \textsc{Evaluate}(H, \mathcal{D}; \theta)$ ▷ (a)
$(H^{\text{best}}, F^{\text{best}}) \leftarrow (\text{Copy}(H), F)$
for $t = 0, \ldots, T-1$ **do**
  $d \leftarrow \textsc{Diagnose}(\mathcal{B}, H)$ ▷ (b)
  $(z, e^{\text{new}}) \leftarrow \textsc{MetaEvolve}(d, \mathcal{E}; \mathcal{O})$ ▷ (c)
  if $z = \text{expand}$ **then** $\mathcal{E} \leftarrow \mathcal{E} \cup \{e^{\text{new}}\}$
  $\boldsymbol{a} \leftarrow \textsc{Edit}(d, \mathcal{E}, e^{\text{new}}; H, \mathcal{S}_t, n)$ ▷ (d)
  if $\boldsymbol{a}$ is empty or invalid **then continue**
  $\widetilde{H} \leftarrow \textsc{Apply}(\text{Copy}(H), \boldsymbol{a})$
  if execution fails **then continue**
  $(\widetilde{F}, \widetilde{\mathcal{B}}) \leftarrow \textsc{Evaluate}(\widetilde{H}, \mathcal{D}; \theta)$
  if $\widetilde{F} \geq F - \tau_{\text{rev}}$ **then**
    $(H, F, \mathcal{B}) \leftarrow (\widetilde{H}, \widetilde{F}, \widetilde{\mathcal{B}})$
    if $F > F^{\text{best}}$ **then**
      $(H^{\text{best}}, F^{\text{best}}) \leftarrow (\text{Copy}(H), F)$
    end if
  end if
end for
return H^{best}

Candidate evaluation and retention. Operations are applied in order to a copy of H and evaluated jointly using the answering pipeline. Execution failure leaves H unchanged. A candidate is retained when its score satisfies $\widetilde{F} \geq F - \tau_{\text{rev}}$, using the incumbent's recorded score F rather than reevaluating it. Harness, score, and diagnostic feedback are replaced together on acceptance; otherwise, the next diagnosis reuses the incumbent's feedback and traces. By preserving newly admitted capabilities regardless of candidate retention, EvoSpace allows subsequent attempts with different arguments. The state-update and rollback rules are detailed in Appendix A.

4 Experiments

We evaluate Ours on two long-term-memory benchmarks, LoCoMo and MemBench, with two LLM backbones. Our main experiments address three questions: (1) *Main results:* Does the complete framework improve answer quality over existing memory systems on both benchmarks and with both backbones? (2) *Evolution trajectory:* Can Ours achieve further gains in later evolution rounds compared with a limited-edit-space baseline? (3) *Ablation study:* Does a dynamically evolving harness edit space outperform limited and fully available spaces, and how do updates to Memory, Retrieval, and Prompt contribute to final performance? Appendix B provides detailed trajectories, recorded edits, and case studies. Appendix B.4 further examines whether evolved Harness updates can be reused across LLM backbones and benchmarks.

4.1 Experimental Setup

Benchmarks We evaluate on two benchmarks covering different interaction regimes:

- **LoCoMo** (Maharana et al., 2024): multi-session dialogues (19–32 sessions per sample, 369–689 turns) with 5 QA categories (single-hop, temporal, multi-hop/inferential, open-domain, adversarial name-swap). We report on the full LoCoMo-10 release: 10 conversations, 1,986 QA pairs.
- **MemBench** (Tan et al., 2025a): a memory-tool-use benchmark with 7 LowLevel categories (simple, comparative, aggregative, conditional, knowledge_update, post_processing, noisy). We evaluate 504 samples drawn as 7 categories $\times$ 2 topics $\times$ 36 samples each.

Metrics & Baselines. LoCoMo uses token-level F1 and BLEU-1. MemBench uses exact-match multiple-choice accuracy. The main tables compare EvolveMem Liu et al. (2026b) and Ours with MemGPT (Packer et al., 2023), Mem0 (Chhikara et al., 2025) , and A-Mem (Xu et al., 2025) on LoCoMo, and RecentMem, MemGPT, and MemBank (Zhong et al., 2024) on MemBench.

Evolution Protocol. For the edit-space comparison, all four settings use the same initial Harness H_0, answer model, and evaluation samples. They share the diagnosis procedure, harness edit selection, candidate execution, and retention rules. R0 is the initial evaluation; evolution proceeds through seven rounds, R1–R7. Each round selects at most $n = 2$ edits in total, allowing combinations across Memory, Retrieval, and Prompt. Diagnosis uses the retained answers, quality scores, and execution traces to identify likely bottlenecks. When a candidate triggers rollback, the entire Harness is restored. We report the performance of the best retained Harness returned at termination, averaged across repeated runs.

Implementation. Ours uses SQLite/FTS5 for storage and BAAI/bge-base-en-v1.5 (768-dim) for embeddings. The initial Harness H_0 uses BM25-only fusion (mode=SUM, semantic and structured views disabled), k_{kw}=5, B_{ctx}=8, with entity-swap and query decomposition disabled, providing a minimal starting point for the self-evolution process. Appendix C details the implementation, including the SQLite schema (Appendix C.1) and embedding configuration (Appendix C.2).

4.2 Main Results

Tables 1 and 2 report the full comparison. Ours achieves the best overall performance on both benchmarks and with both backbones, consistently improving over EvolveMem and the other compared memory systems.

Comparison on LoCoMo. With qwen3.6-flash, Ours improves overall F1 from EvolveMem's 59.40 to 65.13, a gain of 5.73 points, or 9.6% relative, and improves BLEU-1 from 56.70 to 61.00. The largest category-level F1 gains are on Adversarial (+7.70 points) and OpenDomain (+6.50). With GPT-5.1, Ours improves F1 and BLEU-1 over EvolveMem by 13.4% and 14.0% in relative terms, respectively, reaching 65.19 F1 and 61.10 BLEU-1. Its largest F1 improvements occur on OpenDomain (+9.94 points) and Temporal (+9.27). The gains over EvolveMem span all five question categories and both backbones, indicating that the evolved Harness is not tied to a single answer model.

Comparison on MemBench. With qwen3.6-flash, Ours attains 79.80 overall accuracy, exceeding the best baseline, MemBank (65.28), by 14.52 points (22.2% relative), and EvolveMem (65.10) by 14.70 points (22.6%). With GPT-5.1, it improves overall accuracy from 64.88 to 75.20, a 15.9% relative gain. The largest gains concentrate in Reasoning: +21.30 points with qwen3.6-flash and +15.74 with GPT-5.1, corresponding to relative improvements of 38.4% and 29.1%, respectively. Ours also leads on Recall and Robustness for both backbones, showing that its gains span all three reported MemBench capabilities.

4.3 Self-Evolution Trajectory

Figure 3 shows that Ours gains its main advantage through continued improvement in later stages. Both methods can temporarily plateau, and Ours does not lead at every stage. The timing of its largest later gains varies across benchmarks and backbones.

Results on LoCoMo. With qwen3.6-flash, Ours rises from about 55.2 to 63.3 F1 between R3 and R5, while EvolveMem remains near 54.5–54.8. EvolveMem subsequently improves, but Ours retains the higher endpoint. With GPT-5.1, Ours temporarily trails EvolveMem at R3, overtakes it at R4, and reaches 65.19 F1 at R6. These trajectories highlight stronger improvement after plateaus rather than uninterrupted gains.

Results on MemBench. With qwen3.6-flash, the curves largely overlap through R4; Ours then rises from about 65.2 to 78.4 accuracy at R5 while EvolveMem remains near 65. With GPT-5.1, its

Method	MultiHop		SingleHop		Temporal		OpenDomain		Adversarial		Overall	
	F1	BLEU	F1	BLEU	F1	BLEU	F1	BLEU	F1	BLEU	F1	BLEU
qwen3.6-flash												
MemGPT	25.21	23.61	33.69	32.21	24.40	28.05	14.88	17.66	65.79	68.26	31.02	41.94
Mem0	31.86	32.48	37.28	33.15	12.49	21.96	23.38	25.12	77.38	79.12	36.13	49.65
A-Mem	32.40	28.97	35.93	33.67	11.49	17.25	15.54	19.83	67.87	70.27	30.35	42.13
EvolveMem	33.60	34.30	38.40	33.80	57.80	53.60	59.30	55.30	79.70	80.90	59.40	56.70
Ours	**37.50**	**36.20**	**42.90**	**36.50**	**60.20**	**55.60**	**65.80**	**60.00**	**87.40**	**87.80**	**65.13**	**61.00**
Δ with SOTA	+11.61%	+5.54%	+11.72%	+7.99%	+4.15%	+3.73%	+10.96%	+8.50%	+9.66%	+8.53%	+9.65%	+7.58%
GPT-5.1												
MemGPT	28.80	24.30	16.50	14.00	24.90	23.00	29.40	27.30	80.60	80.30	38.31	36.40
Mem0	29.20	24.90	16.00	13.60	26.10	24.20	29.80	27.40	81.90	81.80	38.91	36.90
A-Mem	28.70	24.20	16.40	14.40	24.60	21.80	28.40	26.30	82.60	82.50	38.27	36.50
EvolveMem	35.20	33.90	33.70	35.00	54.10	38.70	51.90	51.00	90.20	87.00	57.47	53.60
Ours	**40.08**	**35.58**	**39.69**	**35.29**	**63.37**	**57.40**	**61.84**	**56.84**	**94.32**	**92.93**	**65.19**	**61.10**
Δ with SOTA	+13.86%	+4.96%	+17.77%	+0.83%	+17.13%	+48.32%	+19.15%	+11.45%	+4.57%	+6.82%	+13.43%	+13.99%

Table 1: Comparison of methods across LoCoMo question categories and backbones using Token-F1 and BLEU-1.

Backbone	Method	Recall	Reasoning	Robustness	Overall
qwen3.6-flash	RecentMem	65.28	50.46	65.28	58.93
	MemGPT	66.67	51.54	65.12	59.52
	MemBank	79.63	54.48	67.13	65.28
	EvolveMem	79.60	55.40	65.00	65.10
	Ours	**86.60**	**76.70**	**77.50**	**79.80**
	Δ with SOTA	+8.75%	+38.45%	+15.45%	+22.24%
GPT-5.1	RecentMem	47.22	20.37	48.15	36.11
	MemGPT	87.50	45.37	62.96	62.43
	MemBank	72.22	47.22	58.80	55.75
	EvolveMem	86.11	54.17	59.72	64.88
	Ours	**90.97**	**69.91**	**67.36**	**75.20**
	Δ with SOTA	+3.97%	+29.06%	+6.99%	+15.91%

Table 2: Comparison of methods across MemBench categories and backbones using accuracy.

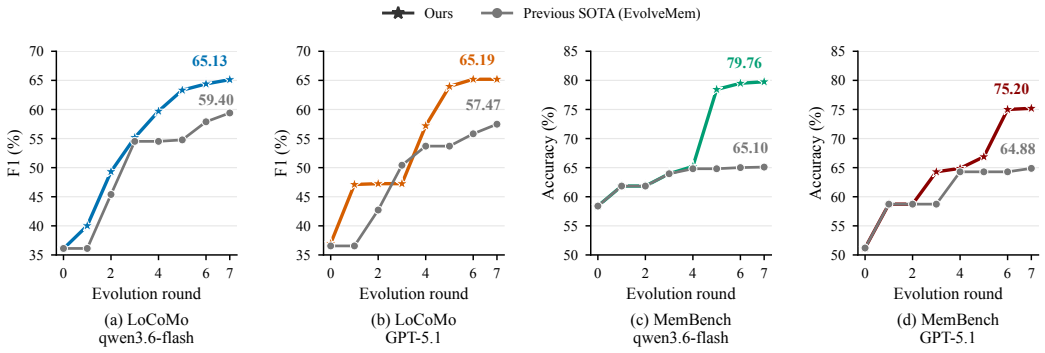


Figure 3: Performance across Harness evolution rounds on (a,b) LoCoMo and (c,d) MemBench with qwen3.6-flash and GPT-5.1.

largest later gain occurs between R5 and R6, from about 66.9 to 75.0. The final plotted gaps are 14.66 and 10.32 points, respectively.

These later gains are consistent with addressing residual failures beyond the initial edit space. When a remaining failure requires an unavailable capability, further tuning within a limited space cannot express the needed repair. EvoSpace uses diagnosis to prioritize the affected component and open a relevant capability, enabling operations that were previously unavailable. This provides a mechanism for renewed improvement after early gains slow. The trajectories alone do not isolate this

Table 3: Edit-space ablation on LoCoMo (F1, %). Qwen: qwen3.6-flash.

Edit space	Qwen	GPT-5.1
Limited $\mathcal{E}_0$	59.40	57.47
Full $\mathcal{E}_{\text{full}}$	58.40	56.51
Scheduled $\mathcal{E}_{\text{sched}}$	62.54	63.32
Dynamic (Ours)	**65.13**	**65.19**

Table 4: Component evolution ablation on LoCoMo with qwen3.6-flash (F1, %). Δ: change from full.

Variant	F1	Δ
Ours (full)	**65.13**	–
w/o PE	47.45	−17.68
w/o ME	55.98	−9.15
w/o RE	58.43	−6.70
w/o PE & ME	43.07	−22.06
w/o PE & RE	45.68	−19.45
w/o RE & ME	48.16	−16.97
w/o PE & ME & RE	36.06	−29.07

effect; Section 4.4 compares limited, fully available, and dynamic spaces to further examine the role of capability admission.

Flat segments may reflect the last committed score carried forward after rollback. Appendix B.1 compares evolution trajectories and documents per-seed edits, adoption and rollback decisions, and capability availability. Appendix B.2 reports the round-by-round overall and category scores, per-seed results, and display conventions.

4.4 Ablation Study

Ablation on Harness Edit-space Evolution. Table 3 compares four settings under the same evolution budget: Limited $\mathcal{E}_0$ retains the initial capabilities; Full $\mathcal{E}_{\text{full}}$ makes all capabilities available from the start; Scheduled $\mathcal{E}_{\text{sched}}$ retains $\mathcal{E}_0$ through R4 and opens Memory repair at R5 without an edit-priority hint; Dynamic (Ours) admits capabilities based on diagnosis and passes the admitted capability to EDIT as a priority hint. Other editing and evaluation rules are shared.

Table 3 shows that broader availability alone is insufficient: Dynamic exceeds Limited and Full by 5.73/7.72 and 6.73/8.68 F1 points on Qwen/GPT-5.1, respectively. Scheduled improves over Limited by 3.14/5.85 points, and Dynamic improves over Scheduled by a further 2.59/1.87 points. Unlike Scheduled, Dynamic also guides EDIT to prioritize the admitted capability, which can favor the diagnosed repair even when admission rounds coincide.

Ablation on Harness Components. Table 4 measures component importance by starting from the full system and removing either one or two Harness update families. Here, "w/o" disables updates to the named components; "w/o PE & ME & RE" disables all three update families and serves as the baseline. Within each group, rows are ordered by the resulting F1 drop.

Table 4 shows that the full system achieves the highest F1 (65.13), 29.07 points above the baseline. Disabling Prompt updates causes the largest single-family drop (17.68 points), versus 9.15 for Memory and 6.70 for Retrieval. Gains are non-additive: Memory and Retrieval together reach 47.45 F1, below the 52.69 obtained by adding their individual gains to the baseline. Thus, in this setting, each update family's benefit depends on the other families available. Appendix B.3 illustrates answer changes following Retrieval, Prompt, and Memory edits.

5 Conclusion

EvoSpace combines shared diagnosis and whole-Harness evaluation to adapt the harness edit space and coordinate Memory, Retrieval, and Prompt updates with a fixed answer model. It consistently outperforms the compared baselines on LoCoMo and MemBench with two backbones. On LoCoMo, adaptive expansion outperforms both fixed and fully available edit spaces under the same evolution budget. Our evaluation covers only memory-grounded QA with implementation-supported edits; future work will explore broader agent tasks and richer editing interfaces.

AI USE STATEMENT

Generative AI tools assisted research ideation and execution, literature search and discovery, and the drafting, organization, and language polishing of this manuscript. Writing assistance was guided by an author-provided outline and experimental summary and included LaTeX preparation and bibliography retrieval. The empirical results reported here originate from the supplied experimental record; no new experiments were performed during manuscript preparation. The authors take full responsibility for the scientific decisions, reported methods and results, citations, and final manuscript.

ETHICS STATEMENT

Conversational memory may contain sensitive personal information. Evidence preservation and provenance improve auditability but also retain source details. Deployments should apply access controls, retention limits, and deletion policies to both original and repaired memories. Query scoping limits retrieval visibility and is not a substitute for privacy protection.

REPRODUCIBILITY STATEMENT

Section 4 specifies the available evaluation protocol, rollout seeds, and aggregate results. Section 3 separates the proposed policy-selection protocol from the observed repair implementation. Appendix D summarizes dependencies, model settings, data and evaluation, and initialization hyperparameters. Complete prompts, raw per-query outputs, search histories, and the full call ledger are not included in the available experimental record, so this draft does not claim complete reproducibility of policy evolution.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective Prompt Evolution Can Outperform Reinforcement Learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2507.19457.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2310.11511.

Qianshu Cai, Yonggang Zhang, Xianzhang Jia, Huajiang Zheng, Wei Xue, Jun Song, Xinmei Tian, and Yike Guo. MOSS: Self-Evolution through Source-Level Rewriting in Autonomous Agent Systems, 2026. URL https://arxiv.org/abs/2605.22794.

Howard Chen, Ramakanth Pasunuru, Jason Weston, and Asli Celikyilmaz. Walking Down the Memory Maze: Beyond Context Limit through Interactive Reading. *arXiv preprint arXiv:2310.05029*, 2023. URL https://arxiv.org/abs/2310.05029.

Mengzhuo Chen, Junjie Wang, Zhe Liu, Yawen Wang, Haiming Zheng, and Qing Wang. From Failed Trajectories to Reliable LLM Agents: Diagnosing and Repairing Harness Flaws, 2026. URL https://arxiv.org/abs/2606.06324.

Ching-An Cheng, Allen Nie, and Adith Swaminathan. Trace is the Next AutoDiff: Generative Optimization with Rich Feedback, Execution Traces, and LLMs, 2024. URL https://arxiv.org/abs/2406.16218. arXiv preprint.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory. In *ECAI 2025*, Frontiers in Artificial Intelligence and Applications. IOS Press, October 2025. ISBN 9781643686318. doi: 10.3233/FAIA251160. URL https://arxiv.org/abs/2504.19413.

Shengran Hu, Cong Lu, and Jeff Clune. Automated Design of Agentic Systems. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://arxiv.org/abs/2408.08435.

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, et al. Memory in the Age of AI Agents. *arXiv preprint arXiv:2512.13564*, 2025b. URL https://arxiv.org/abs/2512.13564.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C. Park. Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models through Question Complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 7036–7050, 2024. doi: 10.18653/v1/2024.naacl-long.389. URL https://arxiv.org/abs/2403.14403.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active Retrieval Augmented Generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7969–7992, 2023. doi: 10.18653/v1/2023.emnlp-main.495. URL https://arxiv.org/abs/2305.06983.

Brandon Kynoch, Hugo Latapie, and Dwane van der Sluis. RecallM: An Adaptable Memory Mechanism with Temporal Understanding for Large Language Models. *arXiv preprint arXiv:2307.02738*, 2023. URL https://arxiv.org/abs/2307.02738.

Kuang-Huei Lee, Xinyun Chen, Hiroki Furuta, John Canny, and Ian Fischer. A Human-Inspired Reading Agent with Gist Memory of Very Long Contexts. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 26396–26415. PMLR, 2024. URL https://arxiv.org/abs/2402.09727.

Yoonho Lee, Joseph Boen, and Chelsea Finn. Feedback Descent: Open-Ended Text Optimization via Pairwise Comparison, 2025. URL https://arxiv.org/abs/2511.07919. arXiv preprint.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-Harness: End-to-End Optimization of Model Harnesses, 2026. URL https://arxiv.org/abs/2603.28052. arXiv preprint.

Joel Lehman, Jonathan Gordon, Shawn Jain, Kamal Ndousse, Cathy Yeh, and Kenneth O. Stanley. Evolution through Large Models. In *Handbook of Evolutionary Machine Learning*, pp. 331–366. Springer, 2023. doi: 10.1007/978-981-99-3814-8_11. URL https://arxiv.org/abs/2206.08896.

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 H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474. Curran Associates, Inc., 2020. URL https://arxiv.org/abs/2005.11401.

Minhua Lin, Zhiwei Zhang, Hanqing Lu, Hui Liu, Xianfeng Tang, Qi He, Xiang Zhang, and Suhang Wang. MemMA: Coordinating the Memory Cycle through Multi-Agent Reasoning and In-Situ Self-Evolution, 2026. URL https://arxiv.org/abs/2603.18718.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. SimpleMem: Efficient Lifelong Memory for LLM Agents. *arXiv preprint arXiv:2601.02553*, 2026a. URL https://arxiv.org/abs/2601.02553.

Jiaqi Liu, Xinyu Ye, Peng Xia, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. EvolveMem:Self-Evolving Memory Architecture via AutoResearch for LLM Agents. *arXiv preprint arXiv:2605.13941*, 2026b. URL https://arxiv.org/abs/2605.13941.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative Refinement with Self-Feedback. In *Advances in Neural Information Processing Systems 36*, NeurIPS 2023, pp. 46534–46594. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2023. doi: 10.52202/075280-2019. URL https://arxiv.org/abs/2303.17651.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating Very Long-Term Conversational Memory of LLM Agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13851–13870, 2024. URL https://arxiv.org/abs/2402.17753.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery, 2025. URL https://arxiv.org/abs/2506.13131. arXiv preprint.

Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing Instructions and Demonstrations for Multi-Stage Language Model Programs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024. URL https://arxiv.org/abs/2406.11695.

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. URL https://arxiv.org/abs/2310.08560.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Xufang Luo, Hao Cheng, Dongsheng Li, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Jianfeng Gao. On Memory Construction and Retrieval for Personalized Conversational Agents. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2502.05589.

Joon Sung Park, Joseph C O'Brien, Carrie J 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. URL https://arxiv.org/abs/2304.03442.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic Prompt Optimization with "Gradient Descent" and Beam Search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7957–7968. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.494. URL https://arxiv.org/abs/2305.03495.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems 36*, NeurIPS 2023, pp. 8634–8652. Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2023. doi: 10.52202/075280-0377. URL https://arxiv.org/abs/2303.11366.

Theodore R. Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L. Griffiths. Cognitive Architectures for Language Agents. *Transactions on Machine Learning Research (TMLR)*, 2024. URL https://arxiv.org/abs/2309.02427.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic Cheatsheet: Test-Time Learning with Adaptive Memory, 2025. URL https://arxiv.org/abs/2504.07952. arXiv preprint.

Haoran Tan, Zeyu Zhang, Chen Ma, Xu Chen, Quanyu Dai, and Zhenhua Dong. MemBench: Towards More Comprehensive Evaluation on the Memory of LLM-based Agents. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 19336–19352, 2025a. doi: 10.18653/v1/2025.findings-acl.989. URL https://arxiv.org/abs/2506.21605.

Zhen Tan, Jun Yan, I-Hung Hsu, Rujun Han, Zifeng Wang, Long Le, Yiwen Song, Yanfei Chen, Hamid Palangi, George Lee, Anand Rajan Iyer, Tianlong Chen, Huan Liu, Chen-Yu Lee, and Tomas Pfister. In Prospect and Retrospect: Reflective Memory Management for Long-term Personalized Dialogue Agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8416–8439, 2025b. doi: 10.18653/v1/2025.acl-long.413. URL https://arxiv.org/abs/2503.08026.

Bing Wang, Xinnian Liang, Jian Yang, Hui Huang, Shuangzhi Wu, Peihao Wu, Lu Lu, Zejun Ma, and Zhoujun Li. SCM: Enhancing Large Language Model with Self-Controlled Memory Framework. *arXiv preprint arXiv:2304.13343*, 2023. URL https://arxiv.org/abs/2304.13343.

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*, 2024. URL https://arxiv.org/abs/2305.16291.

Yu Wang and Xi Chen. MIRIX: Multi-Agent Memory System for LLM-Based Agents. *arXiv preprint arXiv:2507.07957*, 2025. URL https://arxiv.org/abs/2507.07957.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking Chat Assistants on Long-Term Interactive Memory. *arXiv preprint arXiv:2410.10813*, 2024a. URL https://arxiv.org/abs/2410.10813.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversations. In *First Conference on Language Modeling*, 2024b. URL https://arxiv.org/abs/2308.08155.

Chunqiu Steven Xia, Zhe Wang, Yan Yang, Yuxiang Wei, and Lingming Zhang. Live-SWE-agent: Can Software Engineering Agents Self-Evolve on the Fly?, 2025. URL https://arxiv.org/abs/2511.13646.

Peng Xia, Rujun Han, Zifeng Wang, Yanfei Chen, Yufan Zhuang, Yoonho Lee, Chengsong Huang, Han Yu, Zhongying CuiZhu, Yifei Ming, Huaxiu Yao, Burak Gokturk, Tomas Pfister, and Chen-Yu Lee. RRSI: Regularized Recursive Self-Improvement of Agent Harnesses, 2026. URL https://arxiv.org/abs/2609.24972.

Jing Xu, Arthur Szlam, and Jason Weston. Beyond Goldfish Memory: Long-Term Open-Domain Conversation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pp. 5180–5197, 2022. URL https://arxiv.org/abs/2107.07567.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-Mem: Agentic Memory for LLM Agents. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 17577–17604. Curran Associates, Inc., 2025. doi: 10.52202/085713-0593. URL https://arxiv.org/abs/2502.12110.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. Corrective Retrieval Augmented Generation. *arXiv preprint arXiv:2401.15884*, 2024. URL https://arxiv.org/abs/2401.15884.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schütze, Volker Tresp, and Yunpu Ma. Memory-R1: Enhancing Large Language Model Agents to Manage and Utilize Memories via Reinforcement Learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12805–12825, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.583. URL https://arxiv.org/abs/2508.19828.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large Language Models as Optimizers. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.03409.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629.

Haoran Ye, Xuning He, Vincent Arak, Haonan Dong, and Guojie Song. Meta Context Engineering via Agentic Skill Evolution, 2026. URL https://arxiv.org/abs/2601.21557. arXiv preprint.

Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. Agentic Memory: Learning Unified Long-Term and Short-Term Memory Management for Large Language Model Agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21457–21483, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.981. URL https://arxiv.org/abs/2601.01885.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. TextGrad: Automatic "Differentiation" via Text. *arXiv preprint arXiv:2406.07496*, 2024. URL https://arxiv.org/abs/2406.07496.

Guibin Zhang, Haotian Ren, Chong Zhan, Junhao Wang, He Zhu, Wangchunshu Zhou, and Shuicheng Yan. MemEvolve: Meta-Evolution of Agent Memory Systems. In *International Conference on Machine Learning*, 2026a. URL https://arxiv.org/abs/2512.18746.

Hangfan Zhang, Shao Zhang, Kangcong Li, Chen Zhang, Yang Chen, Yiqun Zhang, Lei Bai, and Shuyue Hu. Self-Harness: Harnesses That Improve Themselves, 2026b. URL https://arxiv.org/abs/2606.09498.

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*, 2026c. URL https://arxiv.org/abs/2602.02474.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Tjarko Lange, and Jeff Clune. Darwin Gödel Machine: Open-Ended Evolution of Self-Improving Agents. In *The Fourteenth International Conference on Learning Representations*, 2026d. URL https://arxiv.org/abs/2505.22954.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating Agentic Workflow Generation, 2024. URL https://arxiv.org/abs/2410.10762. arXiv preprint.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic Context Engineering: Evolving Contexts for Self-Improving Language Models. In *The Fourteenth International Conference on Learning Representations*, 2026e. URL https://arxiv.org/abs/2510.04618.

Zeyu Zhang, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Quanyu Dai, 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. doi: 10.1145/3748302. URL https://arxiv.org/abs/2404.13501.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM Agents Are Experiential Learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024. doi: 10.1609/aaai.v38i17.29936. URL https://arxiv.org/abs/2308.10144.

Zibo Zhao, Jijun Shi, Mo Zhou, Zhongyuan Wang, Shifu Bie, Yunfei Zhang, Xuanting Zhou, Xiangyu Wu, Bin Liu, Ruiming Tang, Wenwu Ou, and Kun Gai. RobustSGPO: Search-Space Control for Agent Harness Evolution, 2026. URL https://arxiv.org/abs/2609.09646.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. MemoryBank: Enhancing Large Language Models with Long-Term Memory. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19724–19731, March 2024. ISSN 2159-5399. doi: 10.1609/aaai.v38i17.29946. URL https://arxiv.org/abs/2305.10250.

Yefan Zhou, Yang Li, Zeyu Leo Liu, Semih Yavuz, and Shafiq Joty. Just-in-Time Memory: Learning to Curate Task-Adaptive Memory for LLM Agents, 2026. URL https://arxiv.org/abs/2609.27334.

A Detailed Formulations

We retain the memory representation and retrieval primitives of EvolveMem (Liu et al., 2026b) and restate the relevant formulations below using the Harness state $H_t = (M_t, R_t, P_t)$. These primitives determine how evidence is stored, ranked, and used; the harness edit space $\mathcal{E}_t$ determines which modifications to this state are available. Throughout, θ denotes the fixed answer-model parameters, Retrieval settings belong to R_t, and answer-generation instructions belong to Prompt P_t.

Memory representation. An inherited memory unit has the form

$$m_j = (u_j, \mu_j, \mathbf{e}_j, \boldsymbol{\eta}_j), \qquad \mu_j \in \mathcal{T}, \quad \mathbf{e}_j \in \mathbb{R}^d, \tag{3}$$

where u_j is its text, μ_j its memory type, $\mathbf{e}_j$ its embedding, and $\boldsymbol{\eta}_j$ its metadata, including entities, importance, confidence, and timestamps. We use u_j for memory content to distinguish it from the task category c_i. Extraction initializes M_0 from the conversation histories. Subsequent Memory operators change stored items, while Retrieval operators change how those items are selected. Storage namespaces for users, workspaces, and sessions organize memory access independently of which harness edits are currently available.

Per-view retrieval scores. For query q and memory m_j, the inherited lexical, semantic, and structured views compute

$$s_{\mathrm{kw}}(q, m_j) = \sum_{w \in q} \mathrm{IDF}(w) \frac{\mathrm{tf}(w, u_j)(k_1 + 1)}{\mathrm{tf}(w, u_j) + k_1(1 - b + b|u_j|/\overline{|u|})}, \tag{4}$$

$$s_{\mathrm{sem}}(q, m_j) = \frac{\mathbf{e}_q^\top \mathbf{e}_j}{\|\mathbf{e}_q\| \, \|\mathbf{e}_j\|}, \tag{5}$$

$$s_{\mathrm{str}}(q, m_j) = \sum_{g \in \{\mathrm{persons,locations,entities}\}} \mathbb{1}[\mathrm{extract}_g(q) \cap \boldsymbol{\eta}_{j,g} \neq \emptyset]. \tag{6}$$

Here $\mathrm{tf}(w, u_j)$ is term frequency, $|u_j|$ is the content length, and $\overline{|u|}$ is the average length in the searched collection. The inherited BM25 constants are $k_1 = 1.5$ and $b = 0.75$. The semantic view uses query embedding $\mathbf{e}_q$, while the structured view counts matching metadata fields. Each enabled view $v \in \{\mathrm{kw}, \mathrm{sem}, \mathrm{str}\}$ returns its top-k_v candidates. Our initial $R_0 = \mathrm{BM25}$ uses only the lexical view; the availability of other native views does not mean that they are all active initially.

Fusion and evidence selection. The inherited fusion choices are sum, weighted sum, and reciprocal rank fusion (RRF). For the enabled views $\mathcal{V}$, their relevance scores are

$$s_{\mathrm{fuse}}(q, m_j) = \begin{cases} \sum_{v \in \mathcal{V}} s_v(q, m_j), & \text{sum}, \\ \sum_{v \in \mathcal{V}} w_v \widetilde{s}_v(q, m_j), & \text{weighted sum}, \\ \sum_{v \in \mathcal{V}: m_j \in \mathcal{R}_v(q)} (k_{\mathrm{rrf}} + r_v(m_j))^{-1}, & \text{RRF}, \end{cases} \tag{7}$$

where $\mathcal{R}_v(q)$ is the view's candidate list, $\widetilde{s}_v$ its normalized score, r_v a candidate's rank, and k_{rrf} the smoothing constant. The inherited ranking can further include memory-quality terms:

$$s(q, m_j) = s_{\mathrm{fuse}}(q, m_j) + \lambda_\iota \iota_j + \lambda_{\mathrm{rec}} \, \mathrm{rec}(m_j) + \rho_j, \tag{8}$$

with importance ι_j, recency $\mathrm{rec}(m_j)$, and entity reinforcement ρ_j. The configured context budget limits the memories passed to the answer model. View budgets and weights are arguments of their Retrieval capabilities. The permission to set RRF fusion is a distinct capability; admitting it changes the edit space, while executing it changes the fusion configuration.

Editing capabilities and concrete operations. Following Section 3.1, an editing capability $e = (\chi, o, \mathcal{A}_e)$ specifies a target component, an operation type, and an allowed argument domain. A concrete operation $a = (e, \alpha)$ binds execution arguments $\alpha \in \mathcal{A}_e$. The fixed registry $\mathcal{O}$ contains preimplemented capabilities, and $\mathcal{E}_t$ contains those currently available. The initial interface matches EvolveMem's available Retrieval and Prompt editing capabilities. These include retrieval configuration, answer-instruction revision, and verification; availability does not imply runtime activation.

At most one new capability for the diagnosed component is admitted per round; changing an execution argument does not constitute admission. For example, admitting the previously unavailable "set fusion to RRF" capability expands $\mathcal{E}_t$. Applying an operation using that capability then changes R_t. Once admitted, reusing the capability does not expand the space again. In contrast, changing top-k within an available capability's argument domain changes only the runtime settings. Memory repair is another admissible capability, as illustrated in Appendix B.1. Admission does not itself modify H_t; its capability identifier is passed to EDIT as a priority hint. The registry and each capability's argument domain remain fixed.

For task $x_i = (h_i, q_i)$ with dataset category c_i, write $H_t[x_i] = (M_t[x_i], R_t, P_t[c_i])$. Here $M_t[x_i]$ combines conversation memories with repair entries drawn from the full conversation history and retrievable only for their assigned query. Retrieval configuration is global, and Prompt settings follow the category label supplied by the dataset. In the notation below, $R_t[x_i] = R_t$ and $P_t[x_i] = P_t[c_i]$. Each evolution round selects at most $n = 2$ operations, which may target different components; a batch Memory repair counts as one operation. A single component-level configuration edit may update several parameters or category-specific settings together. These fields do not count as separate edits.

Answer generation and verification. The runtime instructions and settings for answer generation and verification belong to P_t. The capabilities that modify them belong to $\mathcal{E}_t$. In explicit task-conditioned notation, the answering pipeline in Section 3.2 becomes

$$\begin{aligned} E_{i,t} &= \mathrm{Retrieve}(q_i; M_t[x_i], R_t[x_i]), \\ \hat{y}_{i,t} &= \mathrm{LLM}_\theta(q_i, E_{i,t}; P_t[x_i]). \end{aligned} \tag{9}$$

Answer style and format control specify how evidence is presented and how the answer is expressed. When configured, verification reviews the generated answer against that evidence. A Prompt edit can therefore change the answer without changing either the stored memories or the retrieved evidence. The backbone parameters θ remain fixed.

Evaluation records and diagnosis-guided repair. The controller maintains the evaluated state and its per-question records:

$$\mathcal{L}_t = \{(i, q_i, \hat{y}_{i,t}, f_{i,t}, E_{i,t}, \tau_{i,t})\}_{i=1}^{N}, \qquad f_{i,t} = \mathrm{SCORE}(\hat{y}_{i,t}, y_i^\star). \tag{10}$$

Reference answers are used internally by the evaluator and are not passed to diagnosis. Let $F_t = N^{-1} \sum_i f_{i,t}$. On LoCoMo, token-level F1 controls candidate retention and best-state selection; BLEU-1 is reported separately. MemBench uses multiple-choice accuracy. The controller derives $\mathcal{B}_t = \mathrm{Feedback}(\mathcal{L}_t)$, containing correctness indicators and answer traces, and prioritizes samples with $f_{i,t} = 0$. A score of 1 means completely correct; intermediate F1 values are partial matches. Diagnosis receives $\mathcal{B}_t$ and the current Harness context, and selects one component χ_t by jointly interpreting the failures. Missing stored information, missed retrieval evidence, and misuse of retrieved evidence motivate Memory, Retrieval, and Prompt hypotheses, respectively. These hypotheses guide admission and operation generation; they do not impose a predetermined sequence of component updates.

State updates and rollback. The retained state is $(H_t, F_t, \mathcal{L}_t)$. The controller applies a bundle $\boldsymbol{a}_t$ of at most $n = 2$ operations, allowing cross-component combinations, to a snapshot of H_t and evaluates the resulting candidate once. It retains the entire candidate if $\widetilde{F}_{t+1} \geq F_t - \tau_{\mathrm{rev}}$, using the last recorded incumbent score F_t and the nonnegative tolerance inherited from EvolveMem. Otherwise, it preserves the incumbent Harness, score, and records together. The next diagnosis uses feedback derived from those retained records. Candidate–shadow comparisons reported in the case studies are separate from this retention rule. The updated capability space $\mathcal{E}_{t+1}$ survives rollback. The returned state is the best observed retained Harness H^{best}, with ties favoring the earlier state.

B Extended Experimental Results

We provide a round-by-round comparison of Harness evolution, followed by qwen3.6-flash results on LoCoMo and MemBench across three seeds per dataset, recorded harness edits, and source-verified successful cases. We also report transfer results across backbones and benchmarks. All results come from existing experiment records.

B.1 Harness Evolution and Edit-space Changes

Per-seed LoCoMo edits. Table 5 summarizes configuration edits and commit/revert outcomes, followed by Memory updates.

Example: diagnosis followed by multi-view retrieval. Qwen seed 1 rejects R1 at 33.65 F1 and retains 36.12. Its R1 diagnosis reports 951 zero-F1 queries and recommends:

> Switch fusion from keyword_only to rrf to leverage semantic/structured views.

The proposal sets semantic/keyword/structured budgets to 15/12/7 and context budget to 20. R2 adopts the resulting configuration at 45.39 F1.

Memory repair outcomes. Qwen R5–R7 target 391.7, 278.0, and 246.7 queries per seed and add 514.3, 334.0, and 242.0 memories, respectively, averaged across three seeds. All nine R5–R7 Memory proposals are promoted. Candidate-versus-shadow audits also record context changes for non-target queries.

Per-seed MemBench edits. Table 6 lists the selected component-level edits and decisions for each qwen seed.

Configuration outcomes. Seed 1 adopts R1 and rises from 57.34% to 67.66%.

Memory repair. All three qwen runs promote R5 and R6. Their R5 updates target 148/204/176 queries and add 246/350/303 memories; R6 targets 91/128/107 queries and adds 65/111/82 memories, in seed order.

Capability availability on LoCoMo with qwen3.6-flash. The initial set $\mathcal{E}_0$ matches EvolveMem's available capabilities and remains unchanged through R4. Changes to Retrieval and Prompt during these rounds instantiate existing capabilities. Memory repair is admitted at R5 and remains available through R7; subsequent repair calls reuse this capability rather than expanding the edit space again. Retrieval and Prompt capabilities remain available after Memory repair is admitted.

B.2 Round-by-round Overall and Category Results

LoCoMo. Rejected candidates retain the previous incumbent's score. The qwen results cover seeds 1, 2, and 3.

SingleHop, Temporal, MultiHop, OpenDomain, and Adversarial contain 282, 321, 96, 841, and 446 questions, respectively, following the category labels in the experiment implementation. Overall F1 averages queries, not the five category means. Qwen improves from 36.07 to 65.13 F1, with a final three-seed SD of 0.93.

Individual-seed trajectories. Table 8 exposes each seed's overall curve. Qwen R1 has a large cross-seed spread because its first configuration candidate is rejected in seeds 1 and 2 but adopted in seed 3. An increasing mean does not imply that every seed adopted the same harness edit.

Across all three qwen seeds, zero-F1 answers decrease from 871/869/878 at initialization to 389/374/386 after the configuration phase and then to 226/210/243 after R7, in seed order. These counts distinguish eliminating completely incorrect answers from improving the token overlap of already partially correct answers.

Category-specific progress. Using the final category scores reported in Table 1, the R0-to-R7 gains are 23.04, 12.93, 10.73, 20.97, and 63.67 percentage points for SingleHop, Temporal, MultiHop, OpenDomain, and Adversarial, respectively. The largest gain is in Adversarial, whereas MultiHop remains the lowest-scoring category at 37.50. The fresh frozen-configuration baseline before Memory evolution scores 59.39 F1 (three-seed mean). A displayed R4-to-R5 difference therefore spans that re-evaluation, rather than measuring a same-round candidate-minus-shadow repair effect.

Table 5: Selected LoCoMo edits by qwen seed. Multiple category settings within a single Prompt edit share one $\triangle$ symbol.

Round	Seed 1	Seed 2	Seed 3
R0	$\square$	$\square$	$\square$
R1	X: $\diamond_f$ $\triangle_v$	X: $\diamond_f$ $\triangle_v$	A: $\diamond_{b,f}$
R2	A: $\diamond_{b,f}$	A: $\diamond_{b,f}$	A: $\triangle_{p,5}$
R3	A: $\triangle_{p,5}$	A: $\triangle_{p,5}$	A: $\triangle_{1,2}$
R4	S: $\triangle_{1,2}$	A: $\triangle_{1,2}$	A: $\triangle_4$
R5	A: $\bullet$ $\triangle_3$	A: $\bullet$ $\triangle_3$	A: $\bullet$ $\triangle_3$
R6	A: $\bullet$ $\triangle_4$	A: $\bullet$ $\triangle_4$	A: $\bullet$
R7	A: $\bullet$	A: $\bullet$	A: $\bullet$

Table 6: Selected MemBench edits by qwen seed. At R4, the specified Seed 1 update combines fusion-weight tuning and intent planning.

Round	Seed 1	Seed 2	Seed 3
R0	$\square$	$\square$	$\square$
R1	A: $\diamond_{b,f}$	X: $\diamond_f$ $\triangle_v$	X: $\diamond_{f,w}$
R2	X: $\diamond_s$ $\triangle_v$	X: $\diamond_{f,w}$	X: $\diamond_{f,w}$
R3	X: $\triangle_{p,v}$	X: $\diamond_f$ $\triangle_v$	A: $\diamond_f$ $\triangle_p$
R4	A: $\diamond_w$ $\triangle_p$	$\diamond_w$ $\triangle_p$	$\diamond_w$ $\triangle_p$
R5	A: $\bullet$	A: $\bullet$	A: $\bullet$
R6	A: $\bullet$	A: $\bullet$	A: $\bullet$
R7	A: $\bullet$	A: $\bullet$	A: $\bullet$

Shared edit codes for Tables 5–6. Each symbol denotes one component-level edit; multiple subscripts denote fields or categories updated together within that edit.

$\square$: initial BM25 Harness | $\diamond$: Retrieval configuration | $\triangle$: Prompt configuration
$\bullet$: query-scoped Memory repair | b: retrieval views and budgets | f: fusion mode (RRF where specified)
w: fusion weights | s: entity swap | v: answer verification
p: intent planning | 1–5: LoCoMo category-specific Prompt settings

A = adopt; S = soft-adopt; X = revert; – = edit not specified. R0 is initialization and does not consume an edit.

Table 7: LoCoMo token-F1 (%) for qwen3.6-flash. Overall is mean $\pm$ sample SD across three seeds. R7 category scores follow the main-table summary in Table 1; R0–R6 category scores are recorded seed means. Each seed evaluates the same 1,986 queries.

Round	Overall	SingleHop	Temporal	MultiHop	OpenDomain	Adversarial
R0	36.07 ± 0.13	19.86	47.27	26.77	44.83	23.73
R1	40.02 ± 6.73	25.87	50.25	28.68	49.88	25.47
R2	49.32 ± 5.02	35.02	55.35	31.61	57.69	42.04
R3	55.17 ± 0.96	31.29	55.92	28.92	53.55	78.46
R4	59.71 ± 0.55	39.18	58.75	32.40	59.51	79.64
R5	63.32 ± 0.97	41.33	59.75	35.42	64.47	83.62
R6	64.39 ± 1.15	42.29	60.04	36.55	65.14	86.08
R7	65.13 ± 0.93	42.90	60.20	37.50	65.80	87.40

Table 8: LoCoMo qwen3.6-flash F1 (%) by seed. The final mean matches the Overall F1 in Table 1.

Backbone	Seed	R0	R1	R2	R3	R4	R5	R6	R7
qwen3.6-flash	1	36.12	36.12	45.39	54.52	59.40	62.65	63.78	64.90
qwen3.6-flash	2	36.16	36.16	47.59	54.72	60.34	64.43	65.72	66.15
qwen3.6-flash	3	35.92	47.79	54.98	56.28	59.38	62.87	63.68	64.34
	Mean	36.07	40.02	49.32	55.17	59.71	63.32	64.39	65.13

MemBench. We evaluate the 504-query LowLevel subset, with 72 queries in each of seven categories. The metric is multiple-choice accuracy, not LoCoMo's token-F1. Results cover seeds 1–3.

Table 9: MemBench qwen3.6-flash accuracy (%), three seeds. Overall is mean ± sample SD; category values are seed means. The R7 Overall mean rounds to 79.8%, reported as 79.80 in Table 2.

Round	Overall	Simple	Comparative	Aggregative	Conditional	Knowledge Update	Post-processing	Noisy
R0	58.40 ± 0.92	69.44	46.30	43.98	46.76	77.31	63.89	61.11
R1	61.84 ± 5.04	75.46	56.94	46.76	48.61	78.70	64.81	61.57
R2	61.84 ± 5.04	75.46	56.94	46.76	48.61	78.70	64.81	61.57
R3	63.96 ± 4.51	78.24	64.81	46.30	50.00	80.56	64.81	62.96
R4	65.21 ± 5.98	79.17	63.89	49.07	51.85	80.09	65.74	66.67
R5	78.44 ± 3.68	83.80	87.04	59.72	75.93	88.89	77.78	75.93
R6	79.50 ± 3.30	83.80	89.35	63.43	77.31	87.96	79.17	75.46
R7	79.76 ± 2.88	83.80	87.96	63.43	78.70	89.35	79.63	75.46

Recall comprises simple and knowledge-update queries; Reasoning comprises comparative, aggregative, and conditional queries; Robustness comprises post-processing and noisy queries. These groups contain 144, 216, and 144 questions, respectively, so the overall score is their query-weighted average. At R7, these groups yield Recall 86.6%, Reasoning 76.7%, and Robustness 77.5% at the one-decimal reporting precision used in Table 2, where trailing zeros give 86.60, 76.70, and 77.50, respectively. The detailed Overall mean of 79.76% likewise corresponds to the reported 79.80%.

Table 10: MemBench qwen3.6-flash accuracy (%) by seed across the complete trajectory. The final mean rounds to the 79.80% reported in Table 2.

Seed	R0	R1	R2	R3	R4	R5	R6	R7
1	57.34	67.66	67.66	67.66	70.83	81.94	82.54	82.54
2	58.93	58.93	58.93	58.93	58.93	74.60	75.99	76.79
3	58.93	58.93	58.93	65.28	65.87	78.77	79.96	79.96
Mean	58.40	61.84	61.84	63.96	65.21	78.44	79.50	79.76

Table 10 shows the full displayed trajectory for each seed. All three seeds achieve higher accuracy than at initialization, reaching 82.54, 76.79, and 79.96%, respectively. Their mean is 79.76% (79.8% at one decimal place, displayed as 79.80% in Table 2).

Category-specific progress. Relative to the separate frozen-configuration re-evaluation (65.08%), qwen improves by 14.42 percentage points at R6. Comparative and Conditional show the largest gains, from 63.89 to 89.35 and from 54.63 to 77.31, respectively. Aggregative improves from 47.69 to 63.43 but remains the weakest category.

B.3 Case Studies

We organize qwen3.6-flash cases by the Harness component changed in the recorded edit: Retrieval, Prompt, or Memory. The examples include global Retrieval configuration, category-specific Prompt changes, and Memory supplementation. All cases use seed 1 of the corresponding dataset and have a logged score transition of $0 \rightarrow 1$: token-F1 for LoCoMo and multiple-choice accuracy for MemBench. Each edit description identifies the applied modification and its target. The cases illustrate observed answer transitions under the listed updates; updates may change multiple settings together.

Retrieval edits: correcting evidence retrieval. Table 11 contrasts two errors corrected after a global retrieval update. On LoCoMo, R2 enables multi-view budgets 15/12/7, context budget 20, RRF, and verification. On MemBench, R1 enables semantic retrieval with budget 15 and RRF. Both cases move from an incorrect answer to a source-supported answer after directly relevant evidence enters the retrieved context.

Table 11: Successful qwen3.6-flash cases after Retrieval, Prompt, and Memory edits (seed 1). All scores improve from $0 \rightarrow 1$: token-F1 for LoCoMo and accuracy for MemBench. After answers match the references. Memory cases compare R5 no-change shadows with candidates.

Harness edit / dataset / stage	Question and answer transition	Error before and evidence after
Retrieval LoCoMo R0 → R2	What did Caroline research? her education and career → adoption agencies	The answer names the wrong research topic. R2 retrieves `M000022`: Caroline is researching adoption agencies to have a family. Source D2:8 states: "Researching adoption agencies."
Retrieval MemBench R0 → R1	What is the position of my mother? B: Real Estate Consultant → D: Real Estate Sales Specialist	R0 retrieves the mother's identity but not her job. R1 also retrieves `M000498`, which states Clara Mason's exact position. The source says: "My mom works as a Real Estate Sales Specialist."
Prompt LoCoMo	Did Melanie make the black and white bowl in the photo? No, she used many colors and patterns. → Yes	The earlier answer confuses a different colorful pottery episode with the pictured bowl. Both contexts contain `M000074`, which identifies Melanie as the maker of the black-and-white bowl. D5:8 confirms: "Yeah, I made this bowl in my class."
Memory LoCoMo R5	What book did Caroline recommend to Melanie? Unknown → Becoming Nicole	The shadow retrieves Caroline's favorite book but abstains. The candidate adds the explicit recommendation to Melanie. D7:11 names "Becoming Nicole" and says "Highly recommend it for sure!"
Memory MemBench R5	Which event lasts longer, Woodcraft Fest or Crafted Fair? A: Woodcraft Fest → C: Crafted Fair	The shadow retrieves both durations but chooses the shorter event. The candidate adds: "Woodcraft Fest lasts for three whole days, while Crafted Fair lasts for a whole week." The answer now correctly compares the durations.

The LoCoMo query is `conv-26-3`. Its earlier context contains adoption advice Caroline gives Melanie, not the direct statement about Caroline's own research. The MemBench query is `FirstAgent/simple/roles/22`; the supporting source is session index 1, turn `sid=38`. These traces show improved evidence availability; the LoCoMo update changes both retrieval and verification.

Prompt edits: correcting evidence use within a question type. The LoCoMo example in Table 11 belongs to OpenDomain. Unlike the Retrieval examples, the decisive black-and-white-bowl fact is already retrieved before the update.

For LoCoMo query `conv-26-101`, the post-update answer agrees with both the source turn and the reference. This illustrates successful evidence use after a category-specific Prompt edit. The decisive fact is present in both contexts, although the retrieved ordering differs.

Memory edits: supplementing source evidence. Table 11 compares R5 candidates with their same-round no-change shadows. In both cases, the earlier answer fails despite retrieving related facts. The repairs make a book-recommendation relation explicit on LoCoMo and place two event durations in a single comparison statement on MemBench.

The LoCoMo query is `conv-26-104`, with source D7:11. The MemBench query is `FirstAgent/comparative/events/143`; its repair-target record labels the failure as an answer gap. The added comparison cites D2:11 and D2:43, and the original MemBench conversation states the durations in session index 1, turns `sid=27` and `sid=43`. Both duration facts are present in the shadow context; the added statement makes their comparison explicit.

B.4 Cross-Backbone and Cross-Benchmark Transfer

In Tables 12 and 13, Baseline denotes the initial Harness configuration at round 0 (R0), evaluated with the target backbone on the target benchmark before any evolution updates.

Table 12: Cross-backbone transfer.

Configuration	qwen3.6-flash	GPT-5.1
Baseline	58.40	64.88
$\mathcal{C}_{\text{qwen}}$	79.80	**69.25**
$\mathcal{C}_{\text{GPT}}$	**78.37**	75.20

Table 13: Cross-dataset transfer.

Configuration	LoCoMo F1	MemBench Acc.
Baseline	36.07	58.40
$\mathcal{C}_L$	65.13	**75.99**
$\mathcal{C}_M$	**42.86**	79.80

Cross-backbone transfer. With MemBench fixed, let $\mathcal{C}_{\text{qwen}}$ and $\mathcal{C}_{\text{GPT}}$ denote configurations evolved with each backbone. Table 12 evaluates each configuration with both answer models; each off-diagonal entry is assessed against the baseline in its target column.

Cross-dataset transfer. With qwen3.6-flash fixed, $\mathcal{C}_L$ denotes a configuration evolved on LoCoMo and $\mathcal{C}_M$ one evolved on MemBench. Table 13 evaluates each on both datasets against the corresponding R0 baseline.

Table 12 shows positive but asymmetric backbone transfer. $\mathcal{C}_{\text{qwen}}$ improves GPT-5.1 by 4.37 points, whereas $\mathcal{C}_{\text{GPT}}$ improves qwen3.6-flash by 19.97 points and falls only 1.43 points short of target-backbone evolution; the reverse gap is 5.95 points.

Table 13 reveals the same directional asymmetry. $\mathcal{C}_L$ improves MemBench by 17.59 points and approaches its in-domain result within 3.81 points, while $\mathcal{C}_M$ improves LoCoMo by only 6.79 points and remains 22.27 points behind $\mathcal{C}_L$.

C Implementation Details

C.1 SQLite Schema

We retain EvolveMem's SQLite/FTS5 storage backend and core schema (version 6) (Liu et al., 2026b). The database uses write-ahead logging (WAL), `synchronous=NORMAL`, and enabled foreign-key constraints. The main tables separate memory content, search indices, and mutation records:

- `memories` stores each item's identifier, storage scope, type, content, and summary. Source provenance is represented by `source_session_id`, `source_turn_start`, and `source_turn_end`. JSON-encoded fields hold entities, topics, embeddings, tags, and superseded-item identifiers. Additional fields track importance, confidence, reinforcement, access counts, status, and creation, update, access, and expiration times.
- `memories_fts` provides an FTS5 index over content, summaries, entities, and topics for full-text lookup.
- `memory_events` records mutation events, with an event type, memory identifier, storage scope, timestamp, and event details.
- `memory_links` represents typed relations between memory items, such as `related`, `depends_on`, `elaborates`, and `contradicts`.
- `schema_version` tracks schema migrations.

In the current implementation, vectors are serialized in `embedding_json`; this is a storage representation of the embedding, not a change to the semantic scoring function.

Connection to Harness evolution. The database's `scope_id` identifies the storage namespace and is separate from the target of a harness edit. Our editing interface controls which Memory, Retrieval, or Prompt state an operator may update. A Memory edit operates on the relevant stored

items or targeted Memory patch, whereas a Retrieval or Prompt edit changes the corresponding configuration. Admitting an editing capability expands the available edits without changing the stored memories or database schema. Applying a concrete operation changes the targeted Harness state. The harness edit space $\mathcal{E}_t$ is logically separate from the runtime Harness H_t. Source-turn fields connect repaired evidence to its conversation provenance.

C.2 Embedding Models

As stated in Section 4.1, our experimental embedding model is `BAAI/bge-base-en-v1.5`, which produces 768-dimensional vectors through `sentence-transformers`. The inherited batch encoder uses batches of 32 and normalized embeddings.

Retrieval initialization and evolution. The embedding model supplies the semantic view when that view is enabled; it does not imply semantic retrieval is active from the first round. Our initial Retrieval configuration uses BM25 only, with keyword top-$k = 5$, context budget $B_{\text{ctx}} = 8$, and sum fusion. Semantic and structured views, entity-swap, and query decomposition are initially disabled. Evolution changes retrieval settings by applying a concrete operation $a = (e, \alpha)$ with a Retrieval capability $e \in \mathcal{E}_t$. Retrieval configuration is global in the reported setup; Prompt settings are category-specific, and repaired Memory entries are query-scoped. Changing retrieval arguments does not admit a new capability or change the embedding model. The answer-model parameters remain fixed throughout evolution.

D Reproducibility

Code and dependencies. Our implementation builds on the Python memory and retrieval components of EvolveMem (Liu et al., 2026b), with SQLite/FTS5 storage and `sentence-transformers` for semantic embeddings, as described in Appendix C. Algorithm 1 specifies the harness editing interface, keep-or-expand decision, and state-update control flow.

Models. The reported experiments use qwen3.6-flash and GPT-5.1 as answer backbones, and `BAAI/bge-base-en-v1.5` for 768-dimensional memory embeddings. The answer-model parameters remain fixed: evolution changes the external Harness and its available edits rather than training the backbone.

Data and evaluation. LoCoMo (Maharana et al., 2024) and MemBench (Tan et al., 2025a) are publicly available. Our reported evaluation covers all 10 LoCoMo conversations (1,986 question–answer pairs) and 504 MemBench samples selected as 7 categories $\times$ 2 topics $\times$ 36 samples. LoCoMo is evaluated using token-level F1 and BLEU-1, and MemBench using exact-match multiple-choice accuracy, following Section 4.1. The same query pool supplies adaptation feedback and reported scores. Diagnosis receives correctness indicators and answer traces, with priority given to zero-score samples; reference answers remain internal to the evaluator. Aggregate scores determine candidate retention.

Initialization and hyperparameters. We initialize $H_0 = (M_0, \text{BM25}, p_{\text{base}})$ from extracted conversation memories. The initial keyword budget is $k_{\text{kw}} = 5$, the context budget is $B_{\text{ctx}} = 8$, and fusion uses sum mode; semantic retrieval, structured retrieval, entity-swap, and query decomposition are initially disabled. Initially, $\mathcal{E}_0$ permits the same capabilities as EvolveMem, including global Retrieval edits and category-specific Prompt edits; additional capabilities require admission through harness edit-space evolution. Appendix B.2 gives the seed-level trajectories.

E Prompt Catalog

The templates below adapt selected extraction, answering, and diagnosis instructions from EvolveMem (Liu et al., 2026b) to our Harness and harness edit-space interface. These condensed interface templates specify the DIAGNOSE, META-EVOLVE, capability-admission, and EDIT roles in Section 3. Braced placeholders represent runtime inputs.

Memory extraction (adapted from EvolveMem). This template initializes M_0 from conversation windows. It retains the source template's emphasis on explicit entities and detailed factual restatements, while making source attribution explicit.

Memory Extraction: Prompt Template

```
Extract factual memory entries from the supplied conversation.
Preserve names, quantities, event details, and temporal expressions.
Resolve pronouns only when their referents are supported by the
source.  Convert relative dates only when the dated source turn
supports the conversion.

Previous extraction context:  {context}
Dated dialogue with turn identifiers:  {dialogue}

Return a JSON array of entries with lossless_restatement, keywords,
timestamp, persons, locations, entities, and source_turn_ids.  Use
null for an unresolved timestamp.  Do not add unsupported facts or
duplicate an existing entry solely by paraphrasing it.
```

Query decomposition (adapted from EvolveMem). This optional Retrieval capability is inactive in the initial Harness. Its prompt generates search queries, not answers or new memory facts.

Query Decomposition: Prompt Template

```
Split the question into at most {max_subqueries} single-hop
subquestions that together retrieve the evidence needed to answer it.
If it is already single-hop, return the original question.  Preserve
entity and time constraints; do not assume an unknown intermediate
answer.

Question:  {question}
Return only a JSON array of subquestion strings.
```

Evidence-grounded answering (adapted from EvolveMem). The initial Prompt uses the minimal instruction "Please answer the question." The following is a richer Prompt template, not the initial prompt. Task-specific formatting belongs to P_t; the answer agent receives the question and retrieved evidence, not the reference answer.

Evidence-grounded Answering: Prompt Template

```
Answer the question using the supplied evidence.  Preserve the
distinction between different people, events, and dates.  Follow
the answer format without inventing missing supporting facts.

Question:  {question}
Evidence with memory identifiers:  {evidence}
Answer-format instructions:  {format_instructions}

Return JSON with reasoning and answer.  For free-form QA, give a
concise factual answer.  For multiple-choice QA, return one of the
supplied option letters in answer.
```

The format placeholder accommodates LoCoMo's free-form answers and MemBench's option-based answers. Any uncertainty or abstention policy must be specified by the task Prompt rather than inferred from this abbreviated template.

Answer verification (adapted from EvolveMem). When enabled by the effective Prompt, verification reviews the candidate against the same retrieved evidence. It does not create supporting memories.

Answer Verification: Prompt Template

```
Review the candidate answer against the supplied evidence. Check
entity identity, event identity, temporal grounding, and the
required output format. Revise a factual claim only when the
evidence supports the revision. Preserve approximate dates and other
uncertainty present in the evidence.

Question: {question}
Evidence: {evidence}
Candidate answer: {candidate}
Answer-format instructions: {format_instructions}

Return JSON with reasoning and answer.
```

DIAGNOSE: Harness failure localization. Diagnosis receives correctness feedback and answer traces from the retained Harness, with priority given to completely incorrect samples. It cannot access reference-answer text. It selects the single component most worth repairing after jointly interpreting the failures.

DIAGNOSE: Harness Failure Localization

```
Diagnose failures using the supplied correctness indicators and
traces. Distinguish missing stored information, missed retrieval
evidence, and misuse of retrieved evidence. Prioritize completely
incorrect samples. Select one component to repair based on the
combined evidence.

Current Harness summary: {harness}
Correctness feedback and answer traces: {feedback_and_traces}

Return JSON with error_type, supporting_evidence, affected_tasks,
component, and confidence. Cite trace provenance. component is
one of M, R, or P. Do not select concrete operations or change the
Harness. Reference answers are not provided.
```

META-EVOLVE: capability availability. This role prioritizes opening a relevant new capability for the diagnosed component over parameter tuning within existing capabilities.

META-EVOLVE: Keep or Expand

```
Determine whether the diagnosis supports an unopened capability
for the selected component. If so, choose expand, even if existing
capabilities could also be tuned. Otherwise choose keep. Use only
preimplemented capabilities from the registry.

Diagnosis: {diagnosis}
Available capabilities: {edit_space}
Registry with argument domains and scope constraints:
{operator_registry}

Return JSON with decision only: keep or expand. Do not execute
operations or change the Harness.
```

PROPOSE: capability admission. An expand decision opens at most one additional capability. Its argument domain is fixed by the registry; execution values are generated by EDIT.

Admission Proposal: Editing Capability

```
Select one unopened registry capability for the diagnosed component
that addresses the failure. Parameter changes within an existing
```

```
capability do not count as expansion. Do not create capabilities or
enlarge their argument domains.

Diagnosis: {diagnosis}
Available capabilities and registry: {edit_space_and_registry}

Return JSON with capability_id and rationale. Return capability_id
= null if no supported admission exists. Do not bind execution
arguments.
```

The controller resolves capability_id against $\mathcal{O}$, verifies component membership and absence from $\mathcal{E}_t$, and admits it as described in Section 3.3. The admitted capability identifier is forwarded to EDIT as its priority hint. Keep or invalid admission supplies a null hint.

EDIT: concrete operation generation. EDIT instantiates at most $n = 2$ operations per round, allowing combinations across components. The operations form one jointly evaluated candidate.

EDIT: Concrete Operation Bundle

```
Generate at most two concrete operations in this round using
available capabilities. Operations may target different components
when supported by the diagnosis. Bind arguments within each
capability's fixed domain and satisfy the registry's scope and
parameter constraints. Retrieval edits apply globally; Prompt edits
apply by dataset category. A batch Memory repair counts as one
operation. Prefer the priority capability when constructing a valid
bundle; its use is not mandatory.

Diagnosis: {diagnosis}
Available capabilities: {edit_space}
Priority capability (null if none): {priority_capability}
Current Harness and permitted conversation histories:
{state_and_sources}

Return JSON with operations and rationale. Each operation contains
capability_id and arguments, including required target identifiers
and source_references. List operations in execution order. Return
operations = null if no valid update can be generated. Do not admit
capabilities.
```