Remembering by Asking: Retrieval-Induced Memory Evolution for LLM Agents

Anonymous ACL submission

Abstract

Long-term memory is essential for LLM agents, yet most systems update memory only when new interactions arrive: dialogues trigger addition, revision, or deletion, while the memory bank remains mostly static once the input stream pauses. This contrasts with human memory, which can reorganize through self-initiated recall and be reshaped by external retrieval cues. We propose RIME (**R**etrieval-**I**nduced **M**emory **E**volution), a framework that evolves an existing memory bank through self-driven evolution and user-query-driven adaptation. In the self-driven path, RIME periodically recalls existing memories without a user query to merge temporally related episodes, reduce redundancy, resolve conflicts, and strengthen entity-relation structures. In the user-query-driven path, retrieval activates memories, entities, relations, and temporal cues for the current question; feedback from recalled evidence then guides evidence-grounded refinement, relation revision, and importance-score adjustment while softly demoting inactive or redundant regions. Both paths share a recall-feedback-update process, making memory more compact, better organized, and easier to retrieve. On LoCoMo, RIME improves over D-Mem from 53.5 to 56.2 F1, 43.1 to 49.1 BLEU, and 76.3 to 81.6 GPT-4o-mini judge score, while reducing stored memories by 60.8%.

1 Introduction

Long-term memory is essential for LLM agents that interact with users over long periods of time (Hu et al., 2025; Zhang et al., 2025). In multi-session dialogues, personal assistants, and long-horizon reasoning scenarios, agents must recall temporally distributed facts, maintain user-specific preferences, and connect new questions to previously observed events. To support persistent agents, prior work has explored memory streams, long-term memory modules, and virtual context management for extending LLM agents beyond short interaction contexts (Park et al., 2023; Zhong et al., 2024; Packer et al., 2023). Beyond simply extending the accessible context, recent agent-memory systems further manage stored experiences through graph-based consolidation, dynamic indexing and linking, semantic compression, adaptive memory distillation, and multimodal entity-centric organization (Chhikara et al., 2025; Xu et al., 2026; Ma et al., 2025; Long et al., 2025). Together, these works establish memory as a key infrastructure for persistent language agents and suggest that long-term memory should be structured rather than merely stored.

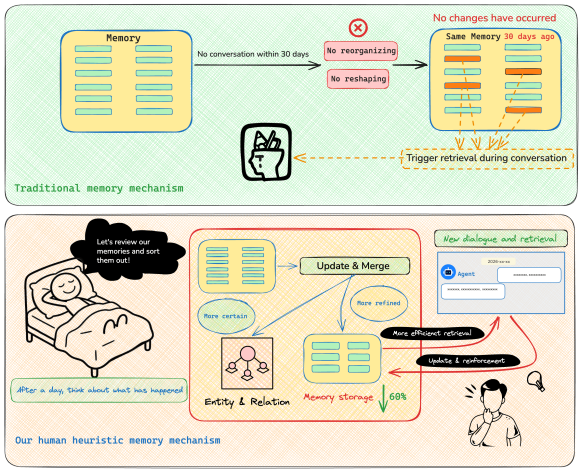

Figure 1: Motivation for memory evolution beyond ingestion. Conventional memory systems update mainly when new dialogues arrive, leaving the memory bank static between interactions. Inspired by human memory, we argue that agent memory should continue to evolve through self-driven consolidation and question-driven refinement, both of which write recall signals back to reinforce useful evidence, strengthen entity-relation structures, and compact redundant traces. Memory should not be a static repository; it should be a living system that evolves through asking.

However, memory evolution in these systems is still largely driven by incoming interactions. In rep-

resentative systems such as Mem0 (Chhikara et al., 2025), new dialogues may trigger summarization, linking, merging, deduplication, compression, or revision of existing entries. Yet such ingestion-time updates do not guarantee an optimized long-term memory structure: the resulting memory bank may still contain fragmented daily traces, partially redundant episodes, unresolved temporal dependencies, and weak entity relations. Once the interaction stream stops, there is usually no mechanism to further evolve these memories, even though temporally related episodes could be merged, redundant traces compacted, and useful entity-temporal links strengthened without new conversations.

As illustrated in Figure 1, this motivates a view of memory evolution beyond ingestion. Memory should not only be updated when new interactions arrive, but also evolve through recall and use. Self-driven recall allows an already constructed memory bank to be revisited and reorganized without a new user query. User-query-driven retrieval, in contrast, activates memories, entities, relations, and temporal cues that are useful for answering a specific question. Both paths expose signals about what should be strengthened, refined, merged, or softly demoted, and these signals should be written back into the memory system rather than discarded after retrieval.

To this end, we propose RIME (**R**etrieval-**I**nduced **M**emory **E**volution), a recall-driven memory framework for LLM agents. RIME first constructs an initial memory bank from raw dialogues using standard memory-management operations, including memory addition, update, deletion, and merging. It then evolves the memory bank through two complementary paths. In the self-driven path, RIME periodically recalls existing memories, organizes related traces, resolves redundancy or conflicts, and strengthens entity-relation structures across time. In the user-query-driven path, RIME uses evolved memory structures to guide retrieval and reranking; after answering, retrieval activation and answer-quality feedback are written back to reinforce useful traces, perform evidence-grounded refinement over recalled memories, update entity-relation scores, and softly demote inactive regions without hard deletion. Thus, both paths follow a similar recall-and-update process to make the memory bank more compact, better organized, and easier to retrieve from over time.

We evaluate RIME on LoCoMo, a benchmark for long-term conversational memory across single-hop, multi-hop, temporal, and open-domain questions (Maharana et al., 2024). RIME improves over the recent state-of-the-art D-Mem (You et al., 2026), increasing F1 from 53.5 to 56.2, BLEU from 43.1 to 49.1, and GPT-4o-mini judge score from 76.3 to 81.6, while reducing stored memories by 60.8%. The gains are especially strong on multi-hop questions, suggesting that entity-relation guided memory evolution helps connect evidence scattered across sessions. These results indicate that long-term agent memory need not simply store more or search harder; it can become more accurate and compact by evolving through recall and use.

Our contributions are summarized as follows:

- **Memory evolution beyond ingestion.** We identify a limitation of ingestion-driven memory systems and formulate long-term memory as an evolving structure that can be reorganized after construction, even without new user interactions.
- **Self-driven and user-query-driven memory evolution.** We propose RIME, a recall-driven framework that combines self-driven consolidation with user-query-driven adaptation to refine recalled memories, update entity-relation scores, and adjust memory-importance scores.
- **Stronger accuracy-memory trade-off on LoCoMo.** RIME improves over D-Mem across F1, BLEU, and LLM-judge evaluation while reducing the memory bank by 60.8%, showing that agent memory can become both more accurate and more compact through use.

2 Related Work

2.1 Memory for LLM Agents.

Memory is essential for LLM agents and has become both a core capability and a major bottleneck, enabling continuity, personalization, experience reuse, and long-horizon task completion (Hu et al., 2025; Zhang et al., 2025; Hatalis et al., 2023). Early agent systems introduce memory streams, long-term user memories, and virtual context management to maintain persistent behavior and personalized interaction (Park et al., 2023; Zhong et al., 2024; Packer et al., 2023). Beyond persistent dialogue, agents also store reflections (Shinn et al., 2023), lessons (Zhao et al., 2024), workflows (Wang et al., 2024), or executable skills (Wang et al.) from previous trials, which can

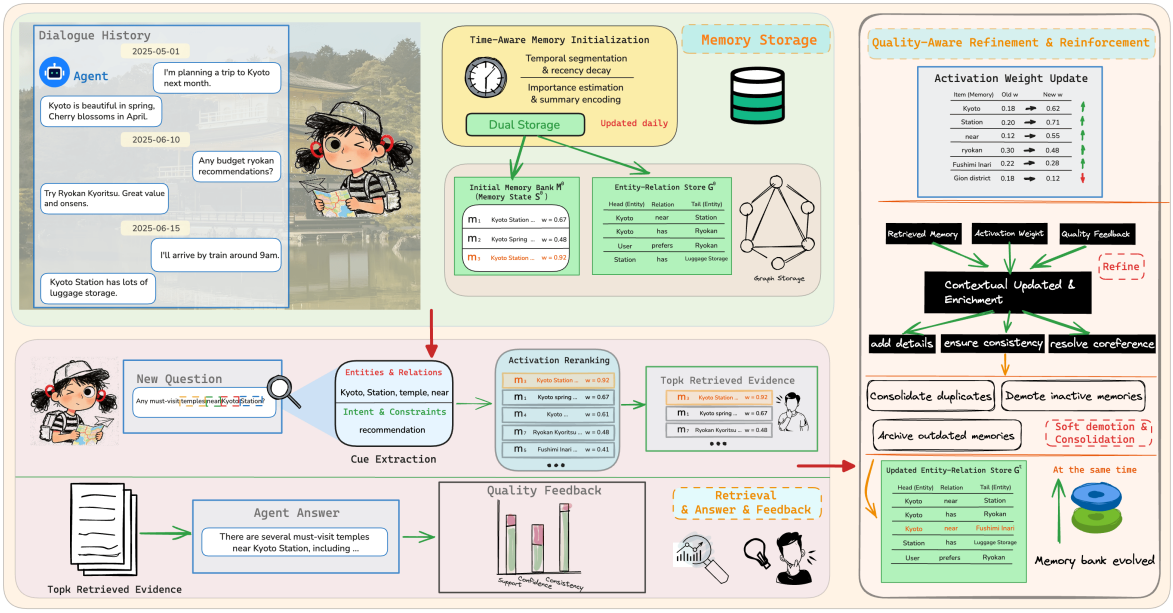

Figure 2: Overview of the RIME recall-feedback-update pipeline. Given dialogue history, RIME first constructs a time-aware memory bank and an entity-relation store. A recall query, which can be either a self-generated consolidation query or an external user question, retrieves relevant memories and entity-relation evidence. The retrieved evidence is used for answer generation or memory consolidation, and an LLM-based critic produces quality feedback. Finally, RIME writes the feedback back to the memory state through content refinement, entity-relation updates, activation-weight adjustment, and memory consolidation, enabling the memory bank to evolve through repeated recall.

be retrieved and inserted into the prompt (Wu et al., 2026) or context to improve future reasoning. However, they mainly study how agents use accumulated memories, whereas we study how memories themselves can continue to evolve.

2.2 Memory Organization and Consolidation.

Memory organization and consolidation aim to make accumulated traces more compact, structured, and retrievable. Prior work manages memories through interaction-triggered operations such as adding new memories, updating existing ones, deleting obsolete entries, and merging related traces, as in Mem0 (Chhikara et al., 2025). Other systems further organize memories through dynamic linking (Xu et al., 2026), semantic compression (Liu et al., 2026), or selective retention of useful experiences (Ma et al., 2025). These methods move beyond raw memory accumulation, but their updates are still largely triggered by incoming experiences: new interactions enter the memory store and then activate linking, compression, consolidation. RIME differs by focusing on recall-driven evolution: self-initiated recall reorganizes stored traces even without new input, while retrieval feedback updates memory importance, entity-relation strengths, and recalled content.

2.3 Retrieval-Time Adaptation and Fallback.

Another line of work improves long-term memory by adapting computation or memory behavior after retrieval. D-Mem combines lightweight retrieval with a Full Deliberation fallback selected by quality gating, improving reliability on LoCoMo when retrieval is uncertain (You et al., 2026; Maharana et al., 2024). Recent studies also show that feedback signals can improve memory behavior, for example through cross-agent trajectory distillation or online self-evolving experience weights (Chang et al., 2026; Zhang et al., 2026). These works support the value of feedback-driven memory updates, but they address different settings such as cross-agent collaboration or continuous task feedback. RIME instead focuses on long-term conversational memory, where self-initiated recall and question-time retrieval feedback jointly evolve an already constructed memory bank.

3 Method

3.1 Overview

RIME aims to make long-term memory evolve after storage. Instead of treating memory as a

static repository that changes only when new dialogues arrive, RIME updates memory through recall events. Each recall event follows a shared pipeline: *trigger* $\rightarrow$ *retrieval* $\rightarrow$ *recalled evidence* $\rightarrow$ *feedback* $\rightarrow$ *memory update*.

Given a raw dialogue history $\mathcal{D}$, RIME maintains an evolving memory state

$$\mathcal{S}^t = \left(\mathcal{M}^t, \mathcal{G}^t, \mathbf{s}_M^t, \mathbf{s}_G^t\right), \tag{1}$$

where $\mathcal{M}^t$ is the memory bank, $\mathcal{G}^t$ is the entity-relation graph, and $\mathbf{s}_M^t$ and $\mathbf{s}_G^t$ denote memory-importance and relation-importance scores. The index t denotes a memory-evolution event rather than a dialogue turn.

A trigger can come from two sources. A *self-driven trigger* is generated by the system itself to revisit, consolidate, and reorganize existing memories. A *user-query trigger* is an external question that retrieves memories for answer generation and then provides feedback to the memory state. Although the two triggers share the same recall-feedback-update interface, they produce different types of feedback. Self-driven feedback focuses on consolidation, coreference resolution, redundancy reduction, conflict handling, and temporal linking. User-query-driven feedback focuses on answer support, confidence estimation, relation usage, and future retrieval priors.

3.2 Time-Aware Memory Initialization

Given a raw dialogue history $\mathcal{D}$, RIME first constructs an initial memory bank $\mathcal{M}^0$. The dialogue stream is processed chronologically, and an LLM-based memory extractor produces memory operations:

$$o_k \in \{\text{add}, \text{update}, \text{delete}, \text{merge}\}. \tag{2}$$

Each memory unit is represented as

$$m_i^0 = (c_i, \tau_i, E_i, R_i, \mathcal{I}_i), \tag{3}$$

where c_i is the memory content, τ_i is the timestamp or session identifier, E_i denotes entities, R_i denotes relations or events, and $\mathcal{I}_i$ stores provenance information.

RIME also initializes an entity-relation graph

$$\mathcal{G}^0 = \{g_1^0, \ldots, g_L^0\}, \tag{4}$$

where each relation is

$$g_l^0 = (h_l, r_l, t_l, \tau_l, \mathcal{I}_l). \tag{5}$$

Algorithm 1 RIME: Recall-Feedback-Update

Require: Initial memory state $\mathcal{S}^0 = (\mathcal{M}^0, \mathcal{G}^0, \mathbf{s}_M^0, \mathbf{s}_G^0)$
for each memory-evolution event t **do**
  Obtain trigger $z_t \in \{z_t^{\text{self}}, z_t^{\text{user}}\}$
  Retrieve memory candidates: $\mathcal{R}_M^t \leftarrow \text{Retrieve}_M(z_t, \mathcal{M}^t, \mathbf{s}_M^t)$
  Retrieve relation candidates: $\mathcal{R}_G^t \leftarrow \text{Retrieve}_G(z_t, \mathcal{G}^t, \mathbf{s}_G^t)$
  Route recalled relations: $(\mathbf{b}^t, \mathcal{G}_{\text{ctx}}^t) \leftarrow \text{Route}(\mathcal{R}_G^t, \mathcal{R}_M^t)$
  Rerank memories with transient relation bonus: $\hat{\mathcal{R}}_M^t \leftarrow \text{Rerank}(\mathcal{R}_M^t, \mathbf{b}^t)$
  Build context: $\mathcal{C}^t \leftarrow \text{Context}(\hat{\mathcal{R}}_M^t, \mathcal{G}_{\text{ctx}}^t)$
  if $z_t = z_t^{\text{self}}$ **then**
    $\mathcal{F}^t \leftarrow \text{Critic}_{\text{self}}(z_t, \mathcal{C}^t)$
  else
    $a_t \leftarrow \text{Answer}(z_t, \mathcal{C}^t)$
    $\mathcal{F}^t \leftarrow \text{Critic}_{\text{user}}(z_t, a_t, \mathcal{C}^t)$
  end if
  $\mathcal{S}^{t+1} \leftarrow \text{Update}(\mathcal{S}^t, \mathcal{F}^t)$
end for
return $\mathcal{S}^T$

Here, h_l and t_l are the head and tail entities or events, r_l is the relation type, τ_l is the temporal cue, and $\mathcal{I}_l$ links the relation to supporting memory IDs. The initial state is only a starting point and will be revised by later recall events.

3.3 Recall-Feedback-Update Pipeline

Given a trigger

$$z_t \in \{z_t^{\text{self}}, z_t^{\text{user}}\}, \tag{6}$$

RIME retrieves both memory evidence and entity-relation evidence:

$$\left(\mathcal{R}_M^t, \mathcal{R}_G^t\right) = \text{Retrieve}\left(z_t, \mathcal{M}^t, \mathcal{G}^t, \mathbf{s}_M^t, \mathbf{s}_G^t\right), \tag{7}$$

where $\mathcal{R}_M^t$ contains recalled memories and $\mathcal{R}_G^t$ contains recalled entity relations.

The recalled relations are used in two ways. If a relation $g_l \in \mathcal{R}_G^t$ is grounded to a recalled memory $m_i \in \mathcal{R}_M^t$ through its provenance IDs, it provides a query-time reranking bonus to m_i. This bonus is transient and is not stored as persistent importance. If a relevant relation is not grounded to the top recalled memories, it is used as additional structured context. We denote the resulting context as

$$\mathcal{C}^t = \text{BuildContext}\left(\mathcal{R}_M^t, \mathcal{R}_G^t\right). \tag{8}$$

For self-driven triggers, the memory critic directly produces consolidation feedback:

$$\mathcal{F}^t_{\text{self}} = \text{Critic}_{\text{self}}\left(z_t, \mathcal{C}^t\right), \tag{9}$$

which identifies coreference links, temporal links, redundant memories, and conflicting memories.

For user-query triggers, RIME first generates an answer from the constructed context:

$$a_t = \text{Answer}\left(z_t, \mathcal{C}^t\right). \tag{10}$$

Then the memory critic produces answer-support feedback:

$$\mathcal{F}^t_{\text{user}} = \text{Critic}_{\text{user}}\left(z_t, a_t, \mathcal{C}^t\right). \tag{11}$$

This feedback records answer confidence, supporting memories, supporting relations, and newly inferred entity relations.

Both feedback types are written in a unified form:

$$\mathcal{F}^t = \left(\mathcal{U}^t_M, \mathcal{U}^t_G, \Delta\mathbf{s}^t_M, \Delta\mathbf{s}^t_G\right), \tag{12}$$

where $\mathcal{U}^t_M$ and $\mathcal{U}^t_G$ denote memory and relation updates, while $\Delta\mathbf{s}^t_M$ and $\Delta\mathbf{s}^t_G$ denote persistent importance-score updates.

Finally, RIME writes the feedback back to the memory state:

$$\mathcal{S}^{t+1} = \text{Update}\left(\mathcal{S}^t, \mathcal{F}^t\right). \tag{13}$$

3.4 Self-Driven Consolidation

Self-driven consolidation provides an intrinsic mechanism for memory evolution. Instead of waiting for new user queries, RIME periodically generates self-driven triggers to revisit the memory state, such as identifying recent events, resolving coreference, checking redundancy, detecting conflicts, and linking temporally related memories.

Given a self-driven trigger, the memory critic produces consolidation feedback. This feedback specifies which memories should be linked, merged or kept as active records. Redundant memories are consolidated into a canonical memory, while their original memory IDs are preserved as provenance. Thus, the canonical memory accumulates stronger supporting evidence rather than simply deleting duplicated traces.

For conflicting memories, RIME distinguishes factual contradiction from temporal state change. If the conflict reflects a state change, the later and better-supported memory is promoted as the active version. Otherwise, the memory with stronger provenance is retained, while weaker conflicting memories are softly demoted.

The resulting score update reflects consolidation quality. Memories receive positive feedback when they absorb redundant evidence, resolve coreference, participate in temporal chains, or become active after conflict resolution.

3.5 User-Query-Driven Adaptation

User-query-driven adaptation is triggered by an external question. Unlike self-driven consolidation, its feedback comes from the answer-generation process. After RIME retrieves memory and relation evidence and generates an answer, the memory critic estimates answer confidence and identifies the memories and relations that support the answer.

This produces answer-support feedback. Memories and relations that contribute to a confident answer receive positive importance updates. The critic may also infer new entity relations from the recalled evidence and insert them into the entity-relation graph with supporting memory IDs.

For example, given the question *How many Prius has Evan owned?*, RIME may recall one memory about Evan's old Prius breaking down and another memory about Evan buying a new Prius. If the answer is *two*, both memories and their corresponding relations are marked as supporting evidence and receive positive feedback. Thus, user queries do not merely consume memory for answering. They also provide usage-based feedback that updates memory importance, relation importance, and entity-relation structure for future recall.

3.6 Feedback-Specific Importance Update and Retrieval Scoring

RIME separates persistent importance from query-time retrieval scores. After each recall event, only feedback-confirmed memories and relations receive persistent updates:

$$\begin{aligned} s^{t+1}_M(i) &= \text{clip}\left(s^t_M(i) + u^t_i\right), \\ s^{t+1}_G(l) &= \text{clip}\left(s^t_G(l) + u^t_l\right). \end{aligned} \tag{14}$$

The memory update depends on the trigger:

$$u^t_i = \begin{cases} u^t_{i,\text{self}}, & z_t \text{ is self-generated}, \\ u^t_{i,\text{user}}, & z_t \text{ is a user query}. \end{cases} \tag{15}$$

Here, $u^t_{i,\text{self}}$ is produced by consolidation feedback, while $u^t_{i,\text{user}}$ is produced by answer-support

feedback. In practice, user-query-driven updates are applied only when the answer confidence exceeds a threshold, e.g., $\mathrm{Conf}(a_t) > 0.8$. Relation scores are updated analogously. RIME uses positive feedback rather than explicit negative penalization: memories that receive less support do not lose importance, but are naturally ranked lower as more useful memories accumulate higher scores.

At retrieval time, each memory is ranked by semantic relevance, persistent memory importance, and relation-grounded evidence:

$$\begin{aligned}\mathrm{Score}(z_t, m_i) =& 0.8 \cdot \mathrm{Sim}(z_t, m_i) + 0.1 \cdot \tilde{s}^t_M(i) \\ &+ 0.1 \cdot \mathrm{RelScore}(z_t, m_i).\end{aligned} \tag{16}$$

Here, $\tilde{s}^t_M(i)$ is the normalized memory importance, and $\mathrm{RelScore}(z_t, m_i)$ aggregates relevant entity relations grounded to m_i through provenance IDs. The relation-based term is query-time only: it affects the current reranking but is not stored as persistent importance. Relevant relations that are not grounded to top-ranked memories are instead passed as structured context for answer generation.

4 Experiments

We evaluate RIME on long-term conversational memory, focusing on three questions: (1) whether memory evolution through retrieval-triggered refinement and query-free consolidation improves answer accuracy, (2) whether it reduces the persistent memory footprint, and (3) which components contribute to the observed gains. Our main hypothesis is that an agent memory system should not only retrieve relevant information, but also become more compact and better organized as its entity-relation structure and memory contents are repeatedly retrieved, refined, and consolidated.

4.1 Experimental Setup

Dataset. We conduct experiments on LoCoMo (Maharana et al., 2024), a benchmark designed to evaluate very long-term conversational memory in LLM agents. LoCoMo consists of 10 long-term multi-session dialogues, with each dialogue containing approximately 24K tokens on average. We evaluate on 1,540 questions covering four major reasoning categories: single-hop, multi-hop, temporal, and open-domain questions. Following our focus on long-term memory retrieval and consolidation, we exclude the adversarial question category from our evaluation.

Baselines. We compare RIME with representative long-term memory methods. The baselines include standard retrieval-augmented memory, Mem0 (Chhikara et al., 2025), a widely used memory extraction baseline, and Nemori (Ma et al., 2025), a recent adaptive memory distillation method. We also compare with D-Mem (You et al., 2026), a recent state-of-the-art system that improves answer quality through a dual-process retrieval mechanism, and include recent memory organization and compression methods such as A-Mem (Xu et al., 2026). For RIME and its ablations, we use GPT-4o-mini as the backbone LLM and follow the same evaluation protocol.

Metrics. We evaluate answer quality using three complementary metrics: F1 (Rajpurkar et al., 2016), BLEU (Papineni et al., 2002), and LLM-as-a-judge evaluation (Zheng et al., 2023), following the evaluation practice used in LoCoMo and recent long-term memory systems (Chhikara et al., 2025; You et al., 2026). F1 and BLEU measure lexical overlap with reference answers, while LLM-as-a-judge captures semantic correctness when valid answers are phrased differently from the reference. We use GPT-4o-mini as the judge model under the official LoCoMo evaluation protocol. We report these metrics for overall performance and for each question category.

To evaluate memory cost, we report the memory compression ratio:

$$\mathrm{Comp.} = 1 - \frac{|\mathcal{M}^T|}{|\mathcal{M}^0|}, \tag{17}$$

where $|\mathcal{M}^0|$ denotes the number of initial memory entries and $|\mathcal{M}^T|$ denotes the number of memory entries after all memory-evolution events during evaluation.

Evaluation Protocol. We evaluate RIME in a sequential online memory setting. During evaluation, user questions are processed in their original order, and each question triggers a user-query-driven recall event for retrieval and answer generation. RIME updates memory using only the current question, recalled evidence, generated answer, and automatic feedback. When a consolidation window ends, RIME additionally generates self-driven recall queries and performs consolidation using only recently accumulated memories and the current memory state. It never accesses gold answers or future questions. For user-query-driven

adaptation, feedback is computed from LLM-based evidence support and consistency signals derived from the recalled memories, rather than from reference answers. This ensures that memory evolution is driven only by information available to the agent at test time. For all methods, final answer quality is evaluated against the official LoCoMo references only after generation.

Method	Answer Quality			Memory
	F1	BLEU	Judge	Comp.
RAG	20.8	16.4	30.2	✗
Mem0	41.5	34.2	61.3	✗
A-Mem	32.4	27.0	52.5	✗
Nemori	50.3	39.7	73.0	✗
D-Mem	53.5	43.1	76.3	✗
RIME	56.2	49.1	81.6	60.8

Table 1: Overall results on LoCoMo. Judge denotes GPT-4o-mini evaluation and Comp. denotes memory compression ratio. Baseline scores are taken from prior reports: RAG, Mem0, A-Mem, and Nemori from Ma et al. (2025), and D-Mem from You et al. (2026). ✗ indicates that comparable persistent memory compression is not reported.

4.2 Overall Accuracy-Memory Trade-off

Table 1 presents the main comparison. RIME achieves the best overall answer quality on LoCoMo while substantially reducing the persistent memory footprint. Compared with D-Mem, the strongest baseline, RIME improves F1 from 53.5 to 56.2, BLEU from 43.1 to 49.1, and the LLM-judge score from 76.3 to 81.6. Meanwhile, RIME compresses the stored memory bank by 60.8%, whereas comparable persistent memory compression ratios are not reported for the baselines. This compression reduces the long-term memory footprint and the retrieval search space from which evidence is selected. It does not assume a smaller answer-generation context: under the same top-k evidence budget, RIME improves answer quality by retrieving from a more compact and better-organized memory state. These results show that RIME improves the accuracy-memory trade-off at the storage and retrieval level, rather than simply reducing the number of tokens.

The key difference between RIME and D-Mem lies in how they improve long-term memory reliability. D-Mem improves answer quality by invoking an additional deliberation stage when lightweight retrieval is insufficient. In contrast, RIME improves the memory state itself: user-query-driven adaptation reinforces useful memories and relations during question answering, while self-driven consolidation periodically reorganizes recent memories and reduces redundancy. The results show that this evolving memory design achieves higher overall answer quality than D-Mem while maintaining a smaller memory bank.

4.3 Performance by Question Type

Table 2 breaks down performance by question type. The most notable gains appear on multi-hop questions, where RIME improves over D-Mem by 7.1 F1 points, 10.4 BLEU points, and 14.2 Judge points. This category requires the system to connect evidence distributed across different sessions, rather than retrieving a single isolated fact. The strong improvement suggests that the entity-relation store is useful not merely as an index, but as a retrieval guide: by following activated entities and relations, RIME can better surface memories that jointly support multi-hop reasoning.

RIME also improves temporal and open-domain questions, though the gains are more moderate. On temporal questions, RIME achieves a comparable F1 score to D-Mem while improving BLEU and Judge, indicating that memory evolution helps preserve temporally relevant evidence without relying on a larger memory bank. On open-domain questions, where the query intent is less constrained and retrieval is more ambiguous, RIME still improves all three metrics. This suggests that quality-aware refinement, activation-aware reranking, and consolidation help reduce noise in the memory state, making useful evidence easier to surface even when the question does not provide a narrow target.

Single-hop questions show smaller but consistent improvements. Since these questions often require one salient memory, the benefit of memory evolution is less dramatic than in multi-hop reasoning. Nevertheless, the gains indicate that reinforcing frequently activated entities and softly demoting inactive memory regions can improve direct factual recall as well. Overall, the category-level results support our central hypothesis: memory evolution is most beneficial when the answer depends on structured connections among entities, relations, and temporally distributed memories.

4.4 Ablation Study

We conduct ablation studies to isolate the contribution of the main components in RIME. The results are shown in Table 3. We compare the full model

Method	Single-Hop			Multi-Hop			Temporal			Open-Domain			Overall		
	F1	BLEU	Judge	F1	BLEU	Judge	F1	BLEU	Judge	F1	BLEU	Judge	F1	BLEU	Judge
RAG	22.2	18.6	32.0	18.6	11.7	31.3	19.5	15.7	23.7	19.0	13.5	32.6	20.8	16.4	30.2
Mem0	44.4	37.7	68.1	34.3	25.2	60.3	44.4	37.6	50.4	27.1	19.4	40.6	41.5	34.2	61.3
A-Mem	35.6	29.2	58.2	24.0	18.8	43.6	38.1	33.8	54.2	9.0	8.6	22.9	32.4	27.0	52.5
Nemori	54.8	43.8	81.9	38.1	26.0	61.7	57.3	47.6	67.6	23.9	18.5	45.8	50.3	39.7	73.0
D-Mem	57.1	47.2	83.0	41.4	29.8	68.8	62.0	50.3	73.2	29.9	22.4	50.0	53.5	43.1	76.3
RIME	59.2	52.6	85.0	48.5	40.2	83.0	62.1	54.9	79.5	32.6	25.8	55.2	56.2	49.1	81.6

Table 2: Performance by question type on LoCoMo. For each category and the overall set, we report F1, BLEU, and GPT-4o-mini Judge scores.

Variant	F1	BLEU	Judge
Full RIME	56.2	49.1	81.6
w/o Ent.-Rel. Guidance	51.4	44.7	74.7
w/o Quality-Aware Update	51.3	44.5	75.2
w/o Self-Driven Consolidation	43.3	37.1	62.1

Table 3: Ablation study of RIME. Ent.-Rel. Guidance denotes entity-relation guided retrieval. Quality-Aware Update removes feedback-based content refinement, relation revision, and importance-score adjustment. Self-Driven Consolidation disables query-free recall and consolidation while keeping user-query-driven adaptation.

with variants that remove entity-relation guidance, quality-aware refinement and reinforcement, and self-driven memory consolidation.

Table 3 shows that all three components contribute to RIME's performance. Removing entity-relation guidance reduces F1 from 56.2 to 51.4, BLEU from 49.1 to 44.7, and Judge from 81.6 to 74.7, showing that structured entity-relation retrieval is important for locating long-term evidence. Removing quality-aware update leads to a similar drop, with F1 decreasing to 51.3, BLEU to 44.5, and Judge to 75.2. This indicates that feedback-guided content refinement, relation revision, and score adjustment help prevent unreliable memory updates. The largest degradation comes from removing self-driven consolidation: F1 drops to 43.3, BLEU to 37.1, and Judge to 62.1. This suggests that RIME's gains are not caused by retrieval reranking alone; compacting and reorganizing the memory state is central to achieving stronger answer quality with a smaller memory footprint.

4.5 Further Analysis

Compression trade-off. We vary the memory retention ratio to study how compactness affects answer quality. This analysis tests whether the observed compression level is a favorable operating point, rather than an overly aggressive reduction of useful evidence.

Retrieval and feedback quality. We inspect retrieved evidence behind final answers and measure supported-answer rate and consistency score from the automatic feedback pipeline. These analyses verify whether entity-relation search and activation-aware reranking surface better evidence, and whether quality feedback prevents unsupported memory updates.

5 Conclusion

We present RIME, a retrieval-induced memory evolution framework that treats memory access as recall queries rather than read-only retrieval. These queries can be self-generated for consolidation or triggered by user questions; both paths write feedback back to memory through content refinement, relation revision, and importance-score adjustment, making the memory bank more compact and better organized.

Experiments on LoCoMo show that RIME improves over D-Mem across F1, BLEU, and LLM-judge evaluation while reducing the memory bank by 60.8%. This suggests that long-term agent memory can improve not by storing more, but by evolving through self-driven consolidation and user-query-driven adaptation.

6 Limitations

This work has three main limitations. First, experiments are limited to LoCoMo; future work should test RIME in broader agent settings such as tool use, planning, and personalization. Second, RIME relies on LLM-based parsing, refinement, and feedback estimation; although we use deterministic decoding and ground updates in retrieved evidence, stronger verification may further improve reliability. Third, memory evolution adds update overhead during interaction, which could be reduced through asynchronous or batched implementation.

References

Yurui Chang, Yiran Wu, Qingyun Wu, and Lu Lin. 2026. Memcollab: Cross-agent memory collaboration via contrastive trajectory distillation. *arXiv preprint arXiv:2603.23234*.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building production-ready AI agents with scalable long-term memory. In *ECAI*, Frontiers in Artificial Intelligence and Applications, pages 2993–3000. IOS Press.

Kostas Hatalis, Despina Christou, Joshua Myers, Steven Jones, Keith Lambert, Adam Amos-Binks, Zohreh Dannenhauer, and Dustin Dannenhauer. 2023. Memory matters: The need to improve long-term memory in llm-agents. In *Proceedings of the AAAI Symposium Series*, volume 2, pages 277–280.

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, and 1 others. 2025. Memory in the age of ai agents. *arXiv preprint arXiv:2512.13564*.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. 2026. Simplemem: Efficient lifelong memory for llm agents. *arXiv preprint arXiv:2601.02553*.

Lin Long, Yichen He, Wentao Ye, Yiyuan Pan, Yuan Lin, Hang Li, Junbo Zhao, and Wei Li. 2025. Seeing, listening, remembering, and reasoning: A multimodal agent with long-term memory. *arXiv preprint arXiv:2508.09736*.

Wenquan Ma, Jiayan Nan, Wenlong Wu, and Yize Chen. 2025. What deserves memory: Adaptive memory distillation for llm agents. *arXiv e-prints*, pages arXiv–2508.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. 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)*, pages 13851–13870.

Charles Packer, Vivian Fang, Shishir_G Patil, Kevin Lin, Sarah Wooders, and Joseph_E Gonzalez. 2023. Memgpt: towards llms as operating systems.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 2383–2392.

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

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*.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. 2024. Agent workflow memory. *arXiv preprint arXiv:2409.07429*.

Yanchen Wu, Tenghui Lin, Yingli Zhou, Fangyuan Zhang, Qintian Guo, Xun Zhou, Sibo Wang, Xilin Liu, Yuchi Ma, and Yixiang Fang. 2026. Memory in the llm era: Modular architectures and strategies in a unified framework. *arXiv preprint arXiv:2604.01707*.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2026. A-mem: Agentic memory for llm agents. *Advances in Neural Information Processing Systems*, 38:17577–17604.

Zhixing You, Jiachen Yuan, and Jason Cai. 2026. D-mem: A dual-process memory system for llm agents. *arXiv preprint arXiv:2603.18631*.

Yaolun Zhang, Yiran Wu, Yijiong Yu, Qingyun Wu, and Huazheng Wang. 2026. Live-evo: Online evolution of agentic memory from continuous feedback. *arXiv preprint arXiv:2602.02369*.

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

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19632–19642.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623.

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

Appendix

A Artifact Licenses and Documentation

We use publicly released scientific artifacts, including the LoCoMo benchmark, previously reported baseline results, and GPT-4o-mini for memory processing and evaluation. We cite the original creators of these artifacts in Sections 1, 2, and 4. We use these artifacts only for research and evaluation purposes, following their published terms of use. We do not redistribute the original datasets, model weights, or proprietary API outputs.

Artifact documentation. LoCoMo is a long-term conversational memory benchmark consisting of multi-session dialogues and questions covering single-hop, multi-hop, temporal, and open-domain reasoning. Section 4.1 reports the number of dialogues, average dialogue length, number of evaluated questions, and evaluated question categories.

B Intended Use

Our use of existing artifacts is consistent with their intended research use: evaluating long-term conversational memory systems. RIME is intended as a research framework for studying memory evolution in LLM agents. It should not be used to infer private attributes, identify individuals, or make high-stakes decisions without additional privacy and reliability safeguards.

C Data Privacy and Content

We use the publicly released LoCoMo benchmark and do not collect new user data. The benchmark is designed for long-term conversational memory evaluation. We do not attempt to identify real individuals, and memory updates are grounded in retrieved benchmark evidence. Future deployments involving real user data should include explicit consent, privacy protection, and mechanisms for removing or correcting stored memories.

D Computational Details

Model and infrastructure. All RIME experiments and ablations use GPT-4o-mini through the OpenAI API for memory parsing, answer generation, feedback estimation, and LLM-as-a-judge evaluation. GPT-4o-mini is a hosted proprietary model, so its exact parameter count and serving infrastructure are not publicly disclosed, and we do not locally deploy or fine-tune the model. Our local computation is limited to data preprocessing, memory construction, retrieval, memory-state updates, and metric computation. These components were run on a server equipped with 8 NVIDIA B200 GPUs; however, GPU acceleration is not required by our implementation because all LLM inference is performed through the API. The experiments can be reproduced on a CPU-only machine with sufficient memory and network access to the API. Since all LLM inference is performed through the API, we report computational cost in terms of API usage rather than GPU-hours. Across development runs, ablations, and final evaluation, all experimental trials used approximately 200 million API tokens in total.

Experimental setup and hyperparameters. All API calls to GPT-4o-mini use deterministic decoding with temperature set to 0, including memory parsing, self-driven consolidation, answer generation, feedback estimation, and LLM-as-a-judge evaluation. Because GPT-4o-mini is accessed through a hosted API, we do not tune model-internal parameters or perform model training. The main controllable hyperparameters are therefore limited to the memory and retrieval components, including the retrieved evidence budget top-k, the weights for semantic relevance, graph relevance, temporal relevance, and activation priority, the consolidation interval, and the memory-update thresholds. These hyperparameters are selected on the development setting and then fixed for all reported main and ablation experiments.

For fair comparison, retrieval uses the same top-k evidence budget across methods whenever results are recomputed, with $k = 30$ in all reported experiments. Ablations use the same decoding settings, retrieval budget, and evaluation protocol as the full RIME model, changing only the component under study.

Result reporting. Unless otherwise noted, all results are averaged over multiple experimental trials conducted under the same evaluation setting. To improve reproducibility and reduce run-to-run variation, we keep prompts, decoding settings, retrieval budgets, API model versions, and update hyperparameters fixed across repeated trials, the main model, and ablations. Tables 1, 2, and 3 report the corresponding averaged scores.

Packages and evaluation. We report token-overlap F1, BLEU, and GPT-4o-mini judge scores

following the Mem0 LoCoMo evaluation protocol. F1 is computed as token-level precision-recall F1 after lowercasing and simple punctuation-based tokenization. BLEU is computed with NLTK's `sentence_bleu` implementation using `SmoothingFunction().method1`; following the evaluation script, we report BLEU-1. All F1, BLEU, and judge scores are reported on a 0–100 scale.

For LLM-as-a-judge evaluation, we use the same prompt format as Mem0 and GPT-4o-mini as the judge model. The judge is run with deterministic decoding, temperature 0.0, and JSON-format output, and assigns each answer a binary correctness label by comparing the generated answer with the gold answer. For baselines whose results are taken from prior work, we report the originally published scores and cite their sources in Section 4.

E Use of AI Assistants

We used AI assistants to support paper writing and editing, including language polishing, clarity improvements, LaTeX formatting suggestions, and checklist preparation. The authors reviewed, edited, and verified all AI-assisted text. AI assistants were not used as a substitute for scientific validation, experimental evaluation, or final author responsibility.