Knowing What is Missing: Efficient Conversational Memory via Explicit Evidence-Gap Tracking

Xingbo Du[1], **Loka Li**[1], **Duzhen Zhang**[1], **Leonard Song**[2], **Le Song**[1]
[1]Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE
[2]The Westminster Schools, Atlanta, GA, USA

Abstract

Long-term conversational memory poses a fundamental system challenge for LLM agents: reasoning over all past interactions improves coverage but incurs prohibitive token costs and noise, while retrieval-based methods are efficient yet often fail on multi-hop and temporal questions. Existing multi-step retrievers partially address this gap, but typically operate over an ever-growing textual history, causing context expansion and noise accumulation across iterations. We propose MemR^3, a backend-agnostic closed-loop controller that reformulates conversational memory retrieval as explicit stateful decision-making. At each iteration, MemR^3 maintains an *evidence-gap state* consisting of grounded information (*evidence*) and unresolved information requirements (*gaps*), and uses this state to route among three actions: `retrieve`, `reflect`, and `answer`. Newly retrieved snippets are incorporated into the state and then masked from subsequent prompts, so the model conditions on a compact summary of progress rather than the full retrieval history. This design turns multi-step memory retrieval into a bounded, inspectable control process whose query-time context scales with the state instead of raw accumulated text. Experiments on LongMemEval$_s$ show that MemR^3 surpasses implicit multi-step retrieval baselines and often approaches or exceeds full-context prompting while retrieving only 1%-5% of the full conversation tokens on long conversations. On LoCoMo, MemR^3 consistently improves over its underlying RAG- and Zep-based memory backbones. These results suggest that explicit evidence-gap tracking is an effective abstraction for token-efficient conversational memory.

1 Introduction

With recent advances in large language model (LLM) agents (Zhong et al., 2023; Li et al., 2026b; Du et al., 2026a), memory systems have become a key component for storing and retrieving long-term, personalized information. They can typically be categorized into two groups: 1) Parametric methods (Wang et al., 2024; Fang et al., 2025) that encode memories implicitly into model parameters, which can handle specific knowledge but struggle with scalability and continual updates, as modifying parameters to incorporate new memories often risks catastrophic forgetting and requires expensive fine-tuning. 2) Non-parametric methods (Xu et al., 2025; LangChain Team, 2025; Chhikara et al., 2025; Rasmussen et al., 2025), in contrast, store explicit external information, enabling flexible retrieval and continual augmentation.

However, as conversational histories grow to hundreds of thousands of tokens, using non-parametric stores exposes a fundamental system challenge: a trade-off between reasoning coverage and context efficiency. Feeding all available memories to the LLM ("Full-Context") provides broad access to evidence but incurs high token cost and can suffer from context dilution (i.e., "lost in the middle"). Conversely, traditional "Retrieve-then-Answer" paradigms (e.g., vanilla RAG) are token-efficient but often miss complex multi-hop and temporal dependencies because retrieval is performed in a

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

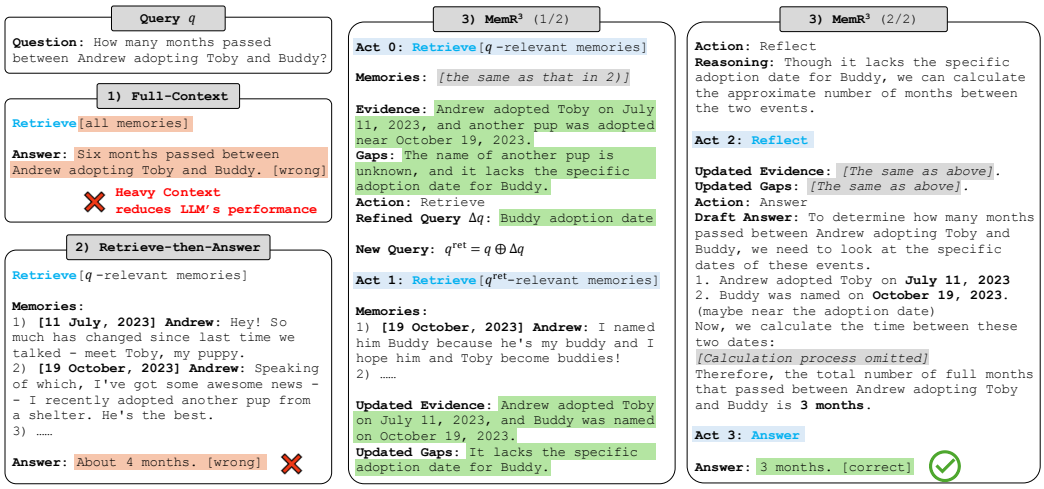

Figure 1: Illustration of three memory-usage paradigms. In this example, **Full-Context** exposes the LLM to many distractors, while **Retrieve-then-Answer** retrieves relevant snippets but still miscalculates. In contrast, **MemR^3** iteratively retrieves and reflects using an explicit evidence-gap tracker (Acts 0-3). By absorbing information and discarding raw texts, it refines the query and produces the correct answer (3 months) without unbounded context expansion.

single open-loop pass. While recent agentic retrieval systems (Asai et al., 2024; Shinn et al., 2023; Yao et al., 2023; Asl et al., 2025) can potentially bridge this gap via multi-step searching, many implementations rely on implicit reasoning traces and append raw retrieved chunks over iterations. In long-term conversational memory, this design causes context expansion and noise accumulation exactly when the agent most needs targeted evidence.

Orthogonal to storage-centric innovations, this paper proposes an agentic memory controller, MemR^3 (i.e., Memory Retrieval system with Reflective Reasoning), designed to improve this trade-off at query time. Instead of relying only on implicit chain-of-thought (Zhang et al., 2025), MemR^3 reformulates multi-step conversational memory retrieval as explicit stateful decision-making. Specifically, the system orchestrates a `router` that dynamically transitions among `retrieve`, `reflect`, and `answer` actions based on an explicit *evidence-gap state*. This state represents the current epistemic status of the agent: grounded information (*evidence*) and unresolved information requirements (*gaps*).

Crucially, MemR^3 employs an "absorb and discard" mechanism. Once newly retrieved snippets are assimilated into the evidence-gap state, their raw texts are masked from future prompts. As a result, key decisions (when to stop, how to refine the next query, and when to reflect) are driven by a compact summary of progress rather than an accumulating context window. This turns multi-step memory retrieval into a bounded and inspectable control process whose query-time context scales with the compact state instead of raw accumulated text.

The system offers three core **advantages**: 1) *Token-Efficient Long-Horizon Reasoning.* By dynamically routing based on explicit gaps and masking assimilated raw snippets, MemR^3 improves reasoning performance without allowing query-time context to grow with the full retrieval trajectory. 2) *Backend-Agnostic Plug-and-Play.* As a stateful controller decoupled from the underlying storage mechanism, MemR^3 can be integrated with existing memory databases (e.g., chunk-based or graph-based) without architectural modifications. 3) *Explicit States and Diagnosability.* By explicitly isolating what is known from what is missing at each step, MemR^3 shifts error attribution from black-box reasoning failures to structural diagnostics (e.g., coverage bias vs. hallucination bias), providing observability into the agent's decision boundary. We compare MemR^3, the Full-Context setting, and the retrieve-then-answer paradigm in Fig. 1. The contributions are threefold:

(1) **A Stateful Closed-Loop Controller.** We propose MemR^3, an autonomous controller that wraps existing memory stores and elevates standard retrieve-then-answer pipelines into a closed-loop process. This instantiates a principled, state-driven paradigm in which routing and query refinement are conditioned on an explicit epistemic state rather than on implicit reasoning.

(2) **Explicit Evidence-Gap Compression.** We formalize an evidence-gap abstraction that summarizes the knowns and unknowns of a given query. This mechanism serves as a stochastic state estimator that compresses retrieval histories into a compact state whose size depends on the current summary rather than the full trajectory, explicitly isolating final reasoning from noisy raw text.

(3) **Empirical Evaluation on LongMemEval$_s$ and LoCoMo.** We evaluate MemR^3 on the long-horizon LongMemEval$_s$ benchmark, where it surpasses implicit multi-step retrieval baselines and often approaches or exceeds Full-Context prompting while retrieving only 1%-5% of the full conversation tokens. Furthermore, diagnostic evaluations on LoCoMo validate its plug-and-play capability across diverse memory backends (RAG and Zep), yielding consistent state-driven improvements over the underlying memory backbones.

2 Related Work

Memory for LLM Agents. Prior work on non-parametric agent memory systems (Wu et al., 2026) spans a wide range of fields, including management and utilization (Du et al., 2025), by storing structured (Rasmussen et al., 2025) or unstructured (Zhong et al., 2024) external knowledge. Specifically, production-oriented agents such as MemGPT (Packer et al., 2023) introduce an OS-style hierarchical memory system that allows the model to page information between context and external storage, and SCM (Wang et al., 2025) provides a controller-based memory stream that retrieves and summarizes past information only when necessary. Additionally, Zep (Rasmussen et al., 2025) builds a temporal knowledge graph that unifies and retrieves evolving conversational and business data. A-Mem (Xu et al., 2025) creates self-organizing, Zettelkasten-style memory that links and evolves over time. Mem0 (Chhikara et al., 2025) extracts and manages persistent conversational facts with optional graph-structured memory. Nemori (Ma et al., 2026) self-organizes long-term memory for LLM agents by segmenting conversations into episodes and learning new knowledge. MIRIX (Wang & Chen, 2025) offers a multimodal memory system with six specialized memory types. LightMem (Fang et al., 2026) proposes a lightweight and efficient memory system inspired by the Atkinson-Shiffrin model.

In this work, we focus on long-term conversational memory, leaving parametric or KV-cache mechanisms (Wang et al., 2024; Zhong et al., 2024; Wei et al., 2025) out of scope. While recent non-parametric systems (e.g., Mem0, A-Mem, and Zep) optimize memory *storage*, they often leave query-time control to open-loop or weakly stateful retrieval. Orthogonal to these storage-centric innovations, MemR^3 acts as a backend-agnostic *stateful controller*. By defining an explicit evidence-gap state, it can layer closed-loop, observable query-time control onto existing memory stores.

Agentic Retrieval-Augmented Generation. RAG (Lewis et al., 2020) established the modern retrieve-then-answer paradigm; subsequent work explored stronger retrievers (Karpukhin et al., 2020; Izacard & Grave, 2021). Self-RAG (Asai et al., 2024), Reflexion (Shinn et al., 2023), and ReAct (Yao et al., 2023) extend retrieval with reflection or iterative reasoning. FAIR-RAG (Asl et al., 2025) refines queries using missing-information signals while retaining an accumulated filtered evidence pool. MemGuide (Du et al., 2026b) uses intent-aligned retrieval and missing-slot tracking to select memories and guide user elicitation. MemR^3 instead uses an explicit evidence-gap state to jointly control query refinement, retrieval, reflection, and stopping. Its "absorb and discard" mechanism masks assimilated raw text, keeping query-time prompts bounded by the current state and the most recent observations rather than by the full retrieval trajectory.

3 Methodology

3.1 From Implicit Reasoning to Explicit Markovian States

Existing agentic retrieval systems (e.g., Self-RAG (Asai et al., 2024), ReAct (Yao et al., 2023)) primarily treat multi-step retrieval as an implicit, autoregressive generation process. In these paradigms, the decision to retrieve, reflect, or answer relies entirely on the implicit chain-of-thought within the Large Language Model (LLM). While effective for short-horizon tasks, this approach is fundamentally non-Markovian: to decide the next action, the LLM must attend to an ever-growing trajectory of interleaved reasoning traces, past queries, and generated text.

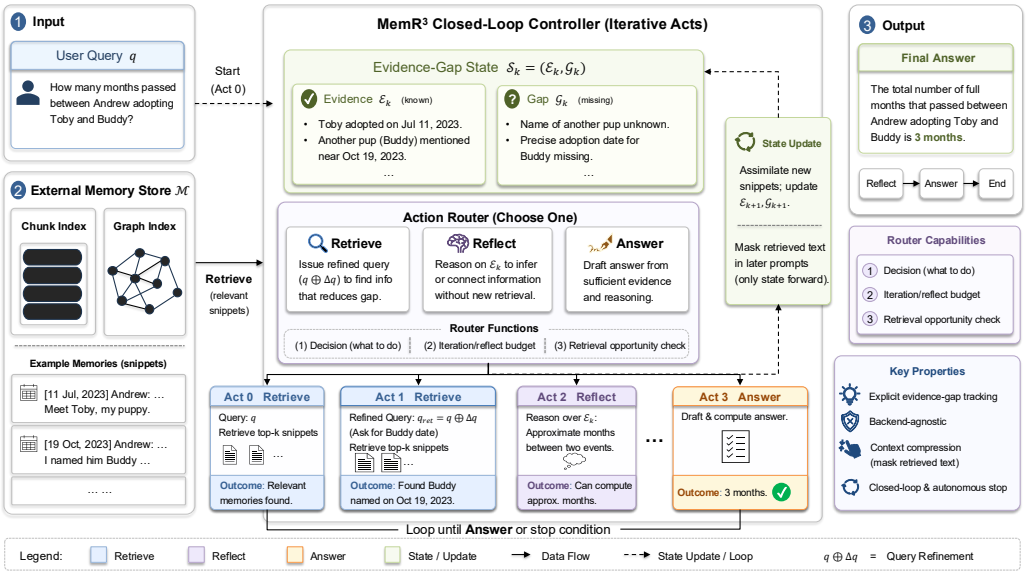

Figure 2: Pipeline of MemR^3. MemR^3 transforms retrieval into a closed-loop process: a router dynamically switches between Retrieve, Reflect, and Answer nodes while a global evidence-gap tracker maintains what is known and what is still missing. This enables iterative query refinement, targeted retrieval, and early stopping, making MemR^3 an autonomous, backend-agnostic controller.

Since the epistemic boundary (what has been resolved vs. what remains missing) is entangled within this unstructured historical log, the agent cannot elegantly compress its past. In long-term memory scenarios, this reliance on an accumulating textual history inevitably leads to context dilution, error propagation (hallucination), and escalating token costs.

To overcome this, we reformulate long-horizon memory retrieval as a Stateful Control Paradigm (SCP). Instead of relying on implicit reasoning over an accumulating textual history, we equip the agent with an explicit epistemic state (belief state) to track its knowledge boundary dynamically. Under an idealized abstraction, this state is Markovian: the next routing decision depends on what has been grounded and what remains missing, rather than on the full textual trajectory. In practice, the state is an LLM-generated approximation, so its reliability must be evaluated empirically and diagnosed through failure modes.

3.2 The Evidence-Gap State Machine: Bridging Theory and Practice

To formalize this stateful control, we first define an idealized state machine and then demonstrate how an LLM can be utilized to instantiate it.

The Latent Requirement Space. For any complex user query q, we conceptualize a latent Information Requirement Space $R(q) = \{r_1, r_2, ..., r_m\}$, which represents the minimal set of atomic facts (e.g., specific entities, timestamps, relations) necessary to completely answer q. The goal of the memory agent is to traverse the memory store $\mathcal{M}$ to fully satisfy $R(q)$.

The Markovian Evidence-Gap Tracker. At any iteration k, the agent maintains an explicit epistemic state $\mathcal{S}_k = (\mathcal{E}_k, \mathcal{G}_k)$.

- **Evidence** ($\mathcal{E}_k \subseteq R(q)$): The set of requirements that have been successfully grounded in the retrieved memory snippets up to step k.
- **Gap** ($\mathcal{G}_k = R(q) \setminus \mathcal{E}_k$): The Dynamic Information Gap Tracker, representing the precise mathematical complement of the evidence. It explicitly defines the epistemic uncertainty, namely, what the agent still needs to know.

In the idealized formulation, $\mathcal{S}_k$ satisfies the Markov property and acts as a sufficient statistic of the retrieval history for the query. The router's policy $\pi(a_k|\mathcal{S}_k)$ then depends only on the current evidence and gap, allowing previously retrieved raw text to be omitted after assimilation. As shown in Theorem 3.1, under an idealized extraction process (detailed in Appendix B), this tracker follows

a strict monotonic update rule: as new memory snippets are retrieved, $\mathcal{E}_k$ monotonically expands, and $\mathcal{G}_k$ monotonically shrinks, until the convergence $\mathcal{G}_k = \emptyset$ is met. The deployable system approximates this behavior with an LLM state estimator; consequently, the guarantees should be read as properties of the target abstraction, not as deterministic guarantees for every prompt execution.

Theorem 3.1 (**Informal**). *Under an idealized requirement space $R(q)$ for a specific query q, the evidence-gap tracker in MemR^3 is **monotone** (evidence never decreases and gaps never increase), **sound** (any requirement supported by the retrieved snippets is added to the evidence set when that support is observed), and **complete** (if the retrieval process eventually surfaces supporting snippets for all requirements in $R(q)$, then the ideal gap becomes empty at convergence).*

In the example in Fig. 2, $\mathcal{E}_k$ may contain "Toby adopted on July 11, 2023" and "Buddy mentioned near October 19, 2023", while $\mathcal{G}_k$ contains "Buddy's precise adoption date is missing". The empirical state stores textual evidence items and unresolved requirements rather than exact symbolic sets.

LLM as a Stochastic State Estimator. In practice, the ideal requirement space $R(q)$ is not analytically given, and exact symbolic matching is intractable for unstructured text. Therefore, we instantiate this theoretical Markov state machine using an LLM. In our MemR^3 system, the LLM serves as a stochastic state estimator.

At each iteration k, the LLM observes the newly retrieved snippets $\mathcal{P}_{k-1}$ and the previous state $(\mathcal{E}_{k-1}, \mathcal{G}_{k-1})$, and stochastically approximates the ideal transition:

$$(\mathcal{E}_k, \mathcal{G}_k) \approx (\mathcal{E}_k^\star, \mathcal{G}_k^\star) = f_{\text{trans}}(\mathcal{P}_{k-1}, \mathcal{E}_{k-1}, \mathcal{G}_{k-1}), \tag{1}$$

where $(\mathcal{E}_k^\star, \mathcal{G}_k^\star)$ is the ideal state, f_{trans} denotes the idealized epistemic update rule-conceptually extracting newly verified facts from the observations $\mathcal{P}_{k-1}$ to monotonically expand the evidence set $\mathcal{E}$ and shrink the remaining gaps $\mathcal{G}$ (as formally defined in Appendix B).

By framing the LLM's prompt-based extraction as a stochastic approximation of an ideal Markov state tracker, MemR^3 provides a principled foundation for its routing mechanism. The router's actions (Retrieve, Reflect, Answer) are conditioned on the estimated state $(\mathcal{E}_k, \mathcal{G}_k)$ together with lightweight control mechanisms, rather than on a growing raw-text log. This decoupling of state estimation and action routing keeps the controller computationally efficient and makes its intermediate decisions inspectable even over extremely long dialogue horizons.

3.3 MemR^3 Architecture: Instantiating the Stateful Controller

To materialize the theoretical Markov state machine into a deployable system, we implement MemR^3 as a directed agentic graph built upon LangGraph (LangChain, 2025). The architecture seamlessly decouples the control logic from the underlying storage, consisting of a central control node (the `Router`) and three operational nodes (`Retrieve`, `Reflect`, `Answer`).

Policy (Router). The LLM updates the evidence-gap state and proposes the next action conditioned on that state. Alg. 1 specifies only deterministic safety overrides. In the implementation, these constraints are encoded as a decision directive before invoking the generator; otherwise, the generator selects among `retrieve`, `reflect`, and `answer` from the estimated state.

Algorithm 1: Safety Overrides for the State-Conditioned Policy

Input: query q, snippets $\mathcal{P}_{k-1}$, iteration k, budgets $n_{\max}, n_{\text{cap}}$, current reflect-streak length n_{reflect}, and original action a_k.
Output: updated action a_k, unchanged if no safety rail is triggered.
if $k \geq n_{\max}$ **then**
  $a_k \leftarrow$ `answer`;
else if $n_{\text{reflect}} \geq n_{\text{cap}}$ **then**
  $a_k \leftarrow$ `retrieve`;
else if $\mathcal{P}_{k-1} = \emptyset$ **then**
  $a_k \leftarrow$ `reflect`;
end if
$k = k + 1$;

Retrieve. When the state indicates missing information, the agent transitions to the `Retrieve` node, generating a precise refinement query Δq_k targeted exclusively at $\mathcal{G}_k$. Crucially, MemR^3 treats the underlying memory store (e.g., chunk-based RAG or graph-based Zep) purely as an observation environment via plug-and-play integration. Once new snippets are assimilated into the state $\mathcal{S}_k$, the raw retrieved texts are explicitly masked from future contexts. This "absorb and discard" mechanism implements the compression discussed earlier. Retrieval excludes previously retrieved snippets by their stable identifiers and searches the remaining candidates. When no additional relevant candidate remains, it returns $\mathcal{P} = \emptyset$, triggering reflection rather than retrieving the same snippets again.

Reflect. The router autonomously transitions to the Reflect node when internal reasoning is deemed more effective than further retrieval. This is common when a retrieval action yields no new information, but the gap remains unresolved. Here, the agent performs internal reasoning to re-evaluate the collected evidence and potentially deduce answers without incurring external retrieval overhead.

Answer. Once the dynamic gap tracker converges ($\mathcal{G}_k = \emptyset$) or the iteration budget is exhausted, the system transitions to the `Answer` node. The final response is generated strictly grounded on the accumulated evidence state $\mathcal{E}_k$. This completely decouples the final reasoning step from the noisy, unbounded raw retrieval context that plagues traditional open-loop RAG systems.

A more detailed illustration of MemR^3 architecture is shown in Appendix C.

3.4 Theoretical and Practical Efficiency

A central dilemma in memory-augmented agents is context accumulation. Multi-step frameworks that append each retrieved chunk to the prompt can incur $O(N \times L)$ context growth over N iterations, where L is the chunk budget per step. This increases cost and can exacerbate "lost in the middle" behavior by flooding the LLM with retrieval noise. MemR^3 mitigates this issue through stateful compression. By employing an "absorb and discard" mechanism, raw retrieved snippets are masked once grounded, carrying forward the semantic state $(\mathcal{E}_k, \mathcal{G}_k)$ and only the newest observations. Consequently, the query-time footprint is bounded by the size of the current evidence-gap state rather than by the full set of accumulated snippets. These architectural choices help MemR^3 retain much of the reasoning robustness of Full-Context prompting while operating closer to the token cost of open-loop RAG.

4 Experiments

We first introduce the experimental setups in Sec. 4.1. Main results include showing the capability of handling long-horizon memory on LongMemEval$_s$ Wu et al. (2025) (Sec. 4.2) and the diagnostic evaluation on LoCoMo (Maharana et al., 2024) (Sec. 4.3). We then present a case study in Sec. 4.4. Other analyses in Sec. 4.5 include ablation studies, efficiency analysis, and the dataset analysis. Code available: `https://github.com/mbzuai-ai4bio/memr3`.

4.1 Experimental Setup

Datasets. We evaluate MemR^3 on two distinct benchmarks: 1) **LongMemEval$_s$** (Wu et al., 2025), a long-horizon dataset comprising 500 questions paired with conversations of approximately 115k tokens on average, serving as the primary testbed for our efficiency-accuracy trade-off claims. 2) **LoCoMo** (Maharana et al., 2024), a fundamental diagnostic benchmark featuring complex question categories (multi-hop, temporal, open-domain, single-hop), averaging 26k tokens per conversation. Following existing work (Chhikara et al., 2025), we exclude the adversarial category.

Baselines as Paradigms. To systematically assess our stateful controller, we categorize baselines into four paradigms: 1) *Full-Memory Reference*: **Full-Context**, which provides a strong reference point when the entire conversation fits the context window (Chhikara et al., 2025). 2) *Open-loop Heuristics*: Vanilla **RAG** (Lewis et al., 2020). 3) *Implicit Agentic Retrievers*: **Self-RAG** (Asai et al., 2024) and a custom strong baseline **RCR** (RAG-CoT-RAG) extended from ReAct (Yao et al., 2023). 4) *Memory Systems*: **A-Mem** (Xu et al., 2025), **LangMem** (LangChain Team, 2025), **Mem0** (Chhikara et al., 2025), and graph-based **Zep** (Rasmussen et al., 2025). We additionally compare with FAIR-RAG (Asl et al., 2025), one-shot query decomposition (Decomp-RAG), and iterative query rewriting (Rewrite-RAG), detailed in Appendix D.2. *Note on Evaluation Scope:* On LongMemEval$_s$, our primary comparison focuses on query-time controllers and full-context prompting. We do not include API-heavy memory-construction systems (e.g., Mem0, Zep) on this benchmark because building explicit indices for 115k-token conversations across all examples incurs substantial ingestion latency and cost. Instead, we include these comparisons on the LoCoMo benchmark.

Metrics & Implementation Details. We adopt the LLM-as-a-Judge (J) score as the metric following Chhikara et al. (2025), because surface-level metrics (e.g., F1, BLEU-1 (Xu et al., 2025; Soni et al., 2024)) can penalize correct open-ended answers that do not lexically match the reference.

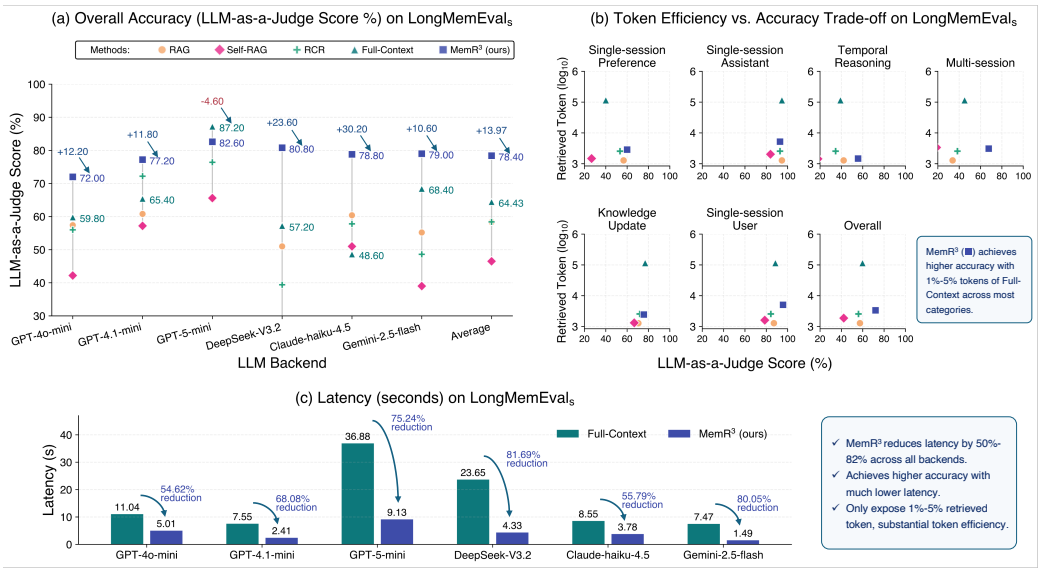

Figure 3: Results on LongMemEval$_s$. (a) Overall accuracy across six LLM backends. (b) Retrieved-context tokens versus accuracy across six categories. (c) LLM-response latency. Repeated-run statistics and full query-time costs are reported in Appendices E.1 and E.4.

Appendix E.1 reports repeated-run statistics, Appendix E.6 provides EM/F1, and Sec. 4.5 reports a human audit of the evidence state. Total input-plus-completion costs are reported in Appendix E.4. Retrieved-context tokens measure evidence volume and are distinct from total LLM usage across controller calls. For LLM backends, we extensively use GPT-4o-mini (OpenAI, 2024b) for baseline tuning and GPT-4.1 (OpenAI, 2025a) to judge whether the answer is correct according to the original question, the generated answer, and the gold answer, following the prompt by Chhikara et al. (2025). For long-context evaluations, we also test other frontier models (GPT-4.1-mini (OpenAI, 2025a), GPT-5-mini (OpenAI, 2025b), DeepSeek-V3.2 (Liu et al., 2025), Claude-haiku-4.5 (Anthropic, 2025), and Gemini-2.5-flash (Comanici et al., 2025)). We set $n_{\text{chk}} = 5$ for RAG and $n_{\max} = 5$ for MemR^3. Detailed tuning processes are in Appendix D.3.

4.2 Improving the Efficiency-Accuracy Trade-off on Long-Horizon Memory

To stress-test our core claim, we evaluate MemR^3 against full-context prompting, one-shot RAG, and implicit multi-step retrievers on the long-horizon LongMemEval$_s$ benchmark across six LLM backends and six categories (Fig. 3).

Limits of Implicit Context Accumulation. The results in Fig. 3 (a) reveal clear limitations of existing retrieval paradigms under long-horizon conversational memory. As expected, open-loop **Vanilla RAG** struggles to capture complex dependencies over sequences of up to 115k tokens. More importantly, implicit multi-step retrievers (**Self-RAG** and **RCR**) often perform worse than vanilla RAG. For example, Self-RAG reaches only 24.00% under DeepSeek-V3.2 and 39.00% under Gemini-2.5-flash. This trend is consistent with our analysis in Sec. 3.4: when raw retrieved chunks are repeatedly appended to the prompt across multiple reasoning steps, the accumulated noise can outweigh the benefits of additional search.

Stateful Compression Improves the Efficiency-Accuracy Trade-off. In Fig. 3 (a), MemR^3 outperforms the retrieval baselines across all six backends and Full-Context on five of six. The exception is GPT-5-mini, where compression can drop temporal and multi-session details that this reasoning backend (OpenAI, 2025b) exploits from the full history (Table 10). On the stronger GPT-5.6 Terra, Full-Context also leads, while consuming about 16-17$\times$ more input tokens (Table 12). Full-Context accuracy thus depends jointly on conversation length and backend capability: very long histories may be difficult for some backends to exploit reliably, whereas stronger long-context backends can use them effectively. MemR^3 therefore targets regimes where full-history inference is costly or unreliable, rather than claiming uniform accuracy superiority over Full-Context; when full-context inference is affordable and reliable, Full-Context remains a strong alternative.

Table 1: Average end-to-end latency (seconds) on LongMemEval$_s$.

Backend	RAG	Self-RAG	RCR	MemR^3	Full-Context
GPT-4o-mini	1.53	7.06	2.81	3.87	11.04
GPT-4.1-mini	1.72	10.71	3.10	5.59	7.55
GPT-5-mini	11.51	41.38	15.80	15.26	36.88

Table 2: LLM-as-a-Judge Score (%) on LoCoMo. Categories are grouped, with GPT-4o-mini (4o) and GPT-4.1-mini (4.1) shown side-by-side. Δ denotes absolute improvements over backbones. MemR^3 rows are three-run means (Appendix E.1); baselines are single runs.

Method	1. Multi-Hop		2. Temporal		3. Open-Domain		4. Single-Hop		Overall	
	4o	4.1	4o	4.1	4o	4.1	4o	4.1	4o	4.1
A-Mem (Xu et al., 2025)	61.70	71.99	64.49	74.77	40.62	58.33	76.63	79.88	69.06	76.00
LangMem (LangChain Team, 2025)	62.23	74.47	23.43	61.06	47.92	67.71	71.12	86.92	58.10	78.05
Mem0 (Chhikara et al., 2025)	67.13	62.41	55.51	57.32	51.15	44.79	72.93	66.47	66.88	62.47
Self-RAG (Asai et al., 2024)	69.15	75.89	64.80	75.08	34.38	54.17	88.31	90.12	76.46	82.08
RCR	71.28	80.85	71.03	81.62	42.71	62.50	86.99	90.12	77.96	84.89
Zep (Rasmussen et al., 2025)	67.38	72.34	73.83	77.26	63.54	64.58	78.67	83.49	74.62	78.94
MemR^3 (Zep)	69.39	77.78	73.83	77.78	**67.01**	69.79	80.60	84.42	76.26	80.88
Δ (abs.)	(+2.01)	(+5.44)	(+0.00)	(+0.52)	(+3.47)	(+5.21)	(+1.93)	(+0.93)	(+1.64)	(+1.94)
RAG (Lewis et al., 2020)	68.79	73.05	65.11	73.52	58.33	62.50	83.86	85.90	75.54	79.46
MemR^3 (RAG)	**71.39**	**81.20**	**76.22**	**82.14**	61.11	**71.53**	**89.44**	**92.17**	**81.55**	**86.75**
Δ (abs.)	(+2.60)	(+8.15)	(+11.11)	(+8.62)	(+2.78)	(+9.03)	(+5.58)	(+6.27)	(+6.01)	(+7.29)
Full-Context	72.34	86.43	58.88	86.82	59.38	71.88	86.39	93.73	76.32	89.00

The advantage of MemR^3 is also reflected in the token-efficiency analysis in Fig. 3 (b). Unlike implicit agentic retrievers that continuously accumulate raw retrieved text in the context window, MemR^3 compresses newly acquired observations into an explicit state and masks assimilated text afterward. As a result, it occupies a more favorable high-accuracy, low-token region across categories, achieving stronger performance while consuming only 1%-5% of the retrieved-context tokens used by Full-Context prompting. These results suggest that long-horizon agents need not choose between one-shot retrieval and brute-force full-context reasoning; stateful compression offers a more effective middle ground.

Latency Reduction. Across all three evaluated backends, MemR^3 is faster than Full-Context and Self-RAG, with additional latency over one-shot RAG from iterative control (Table 1).

4.3 Diagnostic Evaluation on LoCoMo

Having established its long-context efficiency, we utilize the LoCoMo benchmark to explicitly quantify how MemR^3's stateful controller improves complex reasoning across different question typologies and memory backbones (Table 2).

Backend-Agnostic Improvements. Across both LLM backends, MemR^3 consistently improves the performance of its underlying open-loop retrievers (RAG and Zep), supporting its utility as a plug-and-play controller. Most notably, under GPT-4o-mini, the RAG-backed MemR^3 achieves an overall score of 81.55% (+6.01% over vanilla RAG), surpassing all retrieval baselines and Full-Context prompting (76.32%). Under GPT-4.1-mini, MemR^3 yields a +7.29% overall improvement over RAG, suggesting that the benefits of explicit state tracking persist with stronger base models.

Multi-hop & Temporal Reasoning. Multi-hop and temporal questions directly stress a system's ability to coordinate distant memories and avoid noisy over-retrieval. For *temporal queries*, naively providing the entire conversation can hamper reasoning due to timeline confusion, as reflected by Full-Context's 58.88% under GPT-4o-mini. MemR^3 isolates specific timestamps by tracking explicit gaps, reaching 76.22% (+11.11% over RAG). For *multi-hop queries*, MemR^3 explicitly coordinates iterative steps without losing track of intermediate conclusions. Under GPT-4.1-mini, it achieves 81.20% (+8.15% over RAG), slightly exceeding the strong, multi-step RCR baseline (80.85%) without suffering from the latter's context accumulation issues.

Open-Domain & Single-hop. Open-domain questions demand diverse background knowledge, making retrieval harder to trigger accurately. Here, MemR^3 benefits from reformulating queries when initial evidence is noisy. Under GPT-4o-mini, the Zep-backed MemR^3 achieves the highest

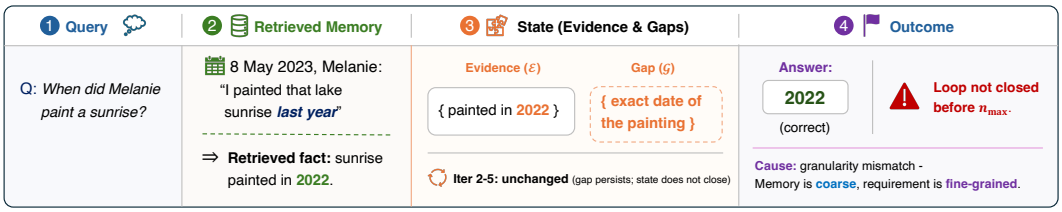

Figure 4: **Granularity bias (failed case).** The agent recovers the coarse fact "in 2022" and produces the correct answer, but the gap requires the exact date. Because of the granularity mismatch, the controller cannot close the loop and stops only at the iteration limit $n_{\max}$.

category score (67.01%), while under GPT-4.1-mini, the RAG-backed variant improves by +9.03% to 71.53%, nearly matching Full-Context (71.88%). For simpler *single-hop* queries, MemR^3 yields strong scores (e.g., 92.17% under GPT-4.1-mini), suggesting that the state-driven early stopping mechanism can terminate queries without over-complicating the reasoning trace.

4.4 Case Study: A Failed Case with Granularity Bias

In Fig. 4, the agent successfully retrieves a coarse-grained fact ("painted in 2022") and produces the correct answer, yet fails to close the loop within the iteration budget. The issue arises from a mismatch between the granularity of the latent requirement and the available evidence: while the memory supports a year-level timestamp, the gap tracker implicitly maintains a finer-grained requirement (e.g., exact date). As a result, the evidence set is sufficient for answering but insufficient for satisfying the residual gap, causing the state to stagnate across iterations. This case highlights that, beyond retrieval coverage or hallucination, granularity mismatch can prevent convergence even when the answer is correct. Importantly, the explicit evidence-gap decomposition makes such failure modes observable: the persistent gap directly reveals that the bottleneck lies in requirement granularity rather than missing retrieval or reasoning errors. The same mismatch bounds open-ended, high-density queries. The state can represent multiple answer-relevant requirements, but selectors such as "major" in "summarize all major life changes" are underspecified and may introduce granularity mismatch or incomplete coverage. Our claim is that state size scales with the selected answer-relevant requirements rather than the full raw history, not that MemR^3 guarantees exhaustive coverage for subjective, high-density summaries.

For other failed cases, such as coverage bias (false negatives) and hallucination bias (false positives), we further present their case studies in Appendix E.8.

4.5 Other analyses (evaluated on LoCoMo)

Table 3: Ablation studies on LoCoMo.

Method	MH*	Temporal	OD*	SH*	Overall
RAG	68.79	65.11	58.33	83.86	75.54
MemR^3	**71.39**	**76.22**	61.11	**89.44**	**81.55**
w/o mask	62.41	68.54	55.21	72.17	68.54
w/o Δq_k	66.67	75.08	60.42	83.37	77.11
w/o reflect	65.25	73.83	**61.46**	83.37	76.65

* MH = Multi-hop; OD = Open-domain; SH = Single-hop. MemR^3: three-run mean.

Ablation Studies. We progressively disable core components of the RAG-backed MemR^3 (Table 3). Disabling the "absorb and discard" masking (*w/o mask*) causes a severe degradation (81.55% $\rightarrow$ 68.54%), consistent with interference from repeatedly exposed raw memories. This variant still answers from the compressed state, so retaining raw history can add interference without recovering omitted evidence. Removing the state-driven refinement query (*w/o* Δq_k) drops single-hop and multi-hop scores significantly, while disabling reflection (*w/o reflect*) removes the agent's ability to reason over existing evidence, harming overall performance (81.55% $\rightarrow$ 76.65%).

State Quality. We manually audit 152 questions from LoCoMo's first conversation under GPT-4o-mini (Table 4). The state preserves answer-critical evidence in 105 of 129 retrieval-success cases (81.4%). Replacing it with a human-verified oracle state built from the same retrieved snippets raises human answer accuracy from 63.2% to 84.2%, identifying state estimation as a remaining bottleneck.

Initial Retrieval Quality. To examine sensitivity to initial retrieval, we count a gold supporting dialogue as covered when retrieved chunks jointly cover at least 50% of its characters. Cumulative

Table 4: Human audit of retrieval and evidence preservation on 152 LoCoMo questions (GPT-4o-mini). $R = 1$: retrieved snippets contain answer-critical evidence; $E = 1$: the final state faithfully retains it.

R	E	Interpretation	Questions	Human accuracy (%)
0	0	Retrieval failure	23	0.0
1	0	State/compression failure	24	4.2
1	1	Evidence preserved	105	90.5

Table 5: Cumulative evidence recall and final J score (%) on LoCoMo, grouped by initial retrieval recall.

Backend	Subset	n	Cumulative recall					J
			Initial	Iter. 2	Iter. 3	Iter. 4	Iter. 5	
GPT-4o-mini	$R_0 = 0$	192	0.00	16.11	24.33	29.77	33.90	45.83
	$0 < R_0 < 1$	206	48.00	52.49	56.92	58.66	59.99	70.87
	$R_0 = 1$	1,124	100.00	100.00	100.00	100.00	100.00	90.04
GPT-4.1-mini	$R_0 = 0$	192	0.00	21.84	34.16	41.50	47.58	64.58
	$0 < R_0 < 1$	206	48.00	54.72	59.54	63.32	65.23	78.64
	$R_0 = 1$	1,124	100.00	100.00	100.00	100.00	100.00	91.99

recall is the fraction of gold dialogues covered by an iteration. Table 5 groups questions by initial recall R_0: iterative retrieval partially recovers initial misses, although initial coverage remains strongly associated with final accuracy.

Revisiting the Evaluation Protocols of LoCoMo. During our reproduction of the baselines, we identified an ambiguity in the LoCoMo dataset's category indexing. Specifically, the mapping between numerical IDs and semantic categories (e.g., Multi-hop vs. Single-hop) requires care because the category names are not directly encoded in the released question files. To avoid conflating categories, we realign the evaluation protocol using category-level question counts and apply the same alignment to all baselines in Table 2. Detailed realignment is illustrated in Appendix F.

5 Conclusion

In this work, we introduce MemR^3, a stateful closed-loop memory controller for long-horizon conversational agents that improves the trade-off between reasoning capability and context efficiency. By modeling multi-step retrieval as explicit evidence-gap tracking, MemR^3 isolates what is known from what remains missing, while its "absorb and discard" mechanism keeps query-time state bounded. Experiments show that MemR^3 improves over its underlying memory backbones, outperforms implicit multi-step retrievers, and often matches or surpasses Full-Context prompting with far fewer tokens. It also makes intermediate memory decisions inspectable, enabling diagnosis of coverage, hallucination, and granularity failures. More capable long-term memory may improve personalized assistants, but may also raise privacy risks if sensitive user information is retrieved.

Limitations. MemR^3 still has several limitations: 1) its performance is bounded by the recall of the underlying memory store; 2) the current system lacks a rollback mechanism, so errors or omissions introduced during state compression may persist after the supporting raw snippets are masked; we leave provenance-guided replay as future work; 3) open-ended requests such as "summarize all major life changes" may suffer from granularity mismatch or incomplete coverage (Sec. 4.4); 4) in LongMemEval$_s$, we do not include ingestion-heavy systems such as Mem0, A-Mem, or Zep due to indexing cost, so we leave large-scale comparisons with such systems to future work; and 5) the current state machine is limited to text-based conversational memory and does not yet extend to multimodal settings (Zhang et al., 2024; Li et al., 2026a).

Acknowledgments and Disclosure of Funding

We thank the anonymous reviewers and the area chair for their constructive comments and suggestions. Le Song is supported by funds from Mohamed bin Zayed University of Artificial Intelligence (MBZUAI) and Weizmann Institute of Science (WIS) projects.

References

Anthropic. Introducing claude haiku 4.5, 2025. URL https://www.anthropic.com/news/claude-haiku-4-5. Accessed: 2025-10-16.

Asai, A., Wu, Z., Wang, Y., Sil, A., and Hajishirzi, H. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=hSyW5go0v8.

Asl, M. A., Asgari-Bidhendi, M., and Minaei-Bidgoli, B. Fair-rag: Faithful adaptive iterative refinement for retrieval-augmented generation. *arXiv preprint arXiv:2510.22344*, 2025.

Chhikara, P., Khant, D., Aryan, S., Singh, T., and Yadav, D. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

Comanici, G., Bieber, E., Schaekermann, M., Pasupat, I., Sachdeva, N., Dhillon, I., Blistein, M., Ram, O., Zhang, D., Rosen, E., et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Du, X., Ghiffari, F. A. A., Song, L., Li, L., Zhang, D., Wang, Z., Chen, X., and Song, L. Biodyad: Synchronize biomedical discovery and machine learning engineering. *arXiv preprint arXiv:2609.31939*, 2026a.

Du, Y., Huang, W., Zheng, D., Wang, Z., Montella, S., Lapata, M., Wong, K.-F., and Pan, J. Z. Rethinking memory in llm based agents: Representations, operations, and emerging topics, 2025. URL https://arxiv.org/abs/2505.00675.

Du, Y., Wang, B., He, Y., Liang, B., Wang, B., Li, Z., Gui, L., Pan, J. Z., Xu, R., and Wong, K.-F. MemGuide: Intent-driven memory selection for goal-oriented multi-session llm agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 30584–30592, 2026b.

Fang, J., Jiang, H., Wang, K., Ma, Y., Shi, J., Wang, X., He, X., and Chua, T.-S. Alphaedit: Null-space constrained knowledge editing for language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Fang, J., Deng, X., Xu, H., Jiang, Z., Tang, Y., Xu, Z., Deng, S., Yao, Y., Wang, M., Qiao, S., Chen, H., and Zhang, N. Lightmem: Lightweight and efficient memory-augmented generation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=dyJ0GWpjJB.

Izacard, G. and Grave, E. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th conference of the european chapter of the association for computational linguistics: main volume*, pp. 874–880, 2021.

Karpukhin, V., Oguz, B., Min, S., Lewis, P. S., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pp. 6769–6781, 2020.

LangChain, I. Langgraph: Build resilient language agents as graphs. https://github.com/langchain-ai/langgraph, 2025.

LangChain Team. Langmem sdk for long-term agent memory. *LangChain Blog*, Feb 2025. URL https://blog.langchain.com/langmem-sdk-launch/. Accessed: 2025-02-18.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

Li, L., Chen, Z., Chen, G., Zhang, Y., Su, Y., Xing, E., and Zhang, K. Confidence matters: Revisiting intrinsic self-correction capabilities of large language models. *arXiv preprint arXiv:2402.12563*, 2024.

Li, L., Kang, W., Fu, M., Chen, G., Chen, Z., Luo, G., Sun, Y., Khan, S., Spirtes, P., and Zhang, K. Personax: Multimodal datasets with llm-inferred behavior traits. In *International Conference on Learning Representations*, volume 2026, pp. 10119–10168, 2026a.

Li, L., Zhang, D., Du, X., Song, L., Wang, Z., Aukenov, A., Thomas, N., Sailaukan, S., Yang, Y., Chen, F., et al. Bioxarena: Benchmarking llm agents on multi-modal biomedical machine learning tasks. *arXiv preprint arXiv:2605.15766*, 2026b.

Liu, A., Mei, A., Lin, B., Xue, B., Wang, B., Xu, B., Wu, B., Zhang, B., Lin, C., Dong, C., et al. Deepseek-v3.2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025.

Ma, W., Nan, J., Wu, W., and Chen, Y. What deserves memory: Adaptive memory distillation for llm agents, 2026. URL https://arxiv.org/abs/2508.03341.

Maharana, A., Lee, D.-H., Tulyakov, S., Bansal, M., Barbieri, F., and Fang, Y. 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.

OpenAI. text-embedding-3-large. https://platform.openai.com/docs/models/text-embedding-3-large, 2024a.

OpenAI. Gpt-4o mini: advancing cost-efficient intelligence, 2024b. URL https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/. Accessed: 2024-07-18.

OpenAI. Introducing gpt-4.1 in the api, 2025a. URL https://openai.com/index/gpt-4-1/. Accessed: 2025-04-14.

OpenAI. Introducing GPT-5, August 2025b. URL https://openai.com/index/introducing-gpt-5/. Accessed: 2025-08-07.

Packer, C., Wooders, S., Lin, K., Fang, V., Patil, S. G., Stoica, I., and Gonzalez, J. E. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Rasmussen, P., Paliychuk, P., Beauvais, T., Ryan, J., and Chalef, D. Zep: a temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956*, 2025.

Reimers, N. and Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36: 8634–8652, 2023.

Soni, A., Arora, R., Kumar, A., and Panwar, D. Evaluating domain coverage in low-resource generative chatbots: A comparative study of open-domain and closed-domain approaches using bleu scores. In *2024 International Conference on Electrical Electronics and Computing Technologies (ICEECT)*, volume 1, pp. 1–6, 2024. doi: 10.1109/ICEECT61758.2024.10738994.

Tang, Z., Chen, Z., Song, X., Li, L., Deng, Y., Shen, Y., Chen, G., Spirtes, P., and Zhang, K. Reflection-window decoding: Text generation with selective refinement. In *International Conference on Machine Learning*, pp. 58739–58764. PMLR, 2025.

Wang, B., Liang, X., Yang, J., Huang, H., Wu, Z., Wu, S., Ma, Z., and Li, Z. Scm: Enhancing large language model with self-controlled memory framework. In *International Conference on Database Systems for Advanced Applications*, pp. 188–203. Springer, 2025.

Wang, P., Li, Z., Zhang, N., Xu, Z., Yao, Y., Jiang, Y., Xie, P., Huang, F., and Chen, H. Wise: Rethinking the knowledge memory for lifelong model editing of large language models. *Advances in Neural Information Processing Systems*, 37:53764–53797, 2024.

Wang, Y. and Chen, X. Mirix: Multi-agent memory system for llm-based agents. *arXiv preprint arXiv:2507.07957*, 2025.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Wei, T., Sachdeva, N., Coleman, B., He, Z., Bei, Y., Ning, X., Ai, M., Li, Y., He, J., Chi, E. H., et al. Evo-memory: Benchmarking llm agent test-time learning with self-evolving memory. *arXiv preprint arXiv:2511.20857*, 2025.

Wu, B., Zhang, Y., Guo, Z., Liu, T., Qian, C., Lu, Y., Du, X., Li, Y., Zhang, Y., Chen, C., et al. Rsmem: Knowledge-enhanced memory evolution for remote sensing agents with systematic evaluation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32899–32915, 2026.

Wu, D., Wang, H., Yu, W., Zhang, Y., Chang, K.-W., and Yu, D. Longmemeval: Benchmarking chat assistants on long-term interactive memory. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=pZiyCaVuti.

Xu, W., Liang, Z., Mei, K., Gao, H., Tan, J., and Zhang, Y. A-mem: Agentic memory for LLM agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=FiM0M8gcct.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. R., and Cao, Y. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Zhang, D., Yu, Y., Dong, J., Li, C., Su, D., Chu, C., and Yu, D. Mm-llms: Recent advances in multimodal large language models. *Findings of the association for computational linguistics: ACL 2024*, pp. 12401–12430, 2024.

Zhang, D., Li, Z.-Z., Zhang, M.-L., Zhang, J., Liu, Z., Yao, Y., Xu, H., Zheng, J., Chen, X., Zhang, Y., et al. From system 1 to system 2: A survey of reasoning large language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

Zhong, R., Du, X., Kai, S., Tang, Z., Xu, S., Zhen, H.-L., Hao, J., Xu, Q., Yuan, M., and Yan, J. Llm4eda: Emerging progress in large language models for electronic design automation. *arXiv preprint arXiv:2401.12224*, 2023.

Zhong, W., Guo, L., Gao, Q., Ye, H., and Wang, Y. MemoryBank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19724–19731, 2024.

A Prompts

A.1 System prompt of the `generate` node

The system prompt is defined as follows, where the "decision_directive" instructs the maximum iteration budget, reflect-streak capacity, and retrieval opportunity check, introduced in Sec. 3.3. Generally, "decision_directive" is a textual instruction: *"reflect" if you need to think about the evidence and gaps; choose "answer" ONLY when evidence is solid and no gaps are noted; choose "retrieve" otherwise.* However, when the maximum iterations budget is reached, "decision_directive" is set as "answer" to stop early. When the reflection reaches the maximum capacity, "decision_directive" is set as "retrieve" to avoid repeated ineffective reflection. When no useful retrieval remains, "decision_directive" is set as "reflect" to avoid repeated ineffective retrieval. Through these constraints, the agent can avoid infinite ineffective actions to maintain stability.

SYSTEM PROMPT

You are a memory agent that plans how to gather evidence before producing the final response shown to the user. Always reply with a strict JSON object using this schema:
- evidence: JSON array of concise factual bullet strings relevant to the user's question; preserve key numbers/names/time references. If exact values are unavailable, include the most specific verified information (year/range) without speculation. Never mention missing or absent information here - "gaps" will do that.
- gaps: gaps between the question and evidence that prevent a complete answer.
- decision: one of ["retrieve","answer","reflect"]. Choose {decision_directive}.

Only include these conditional keys:
- retrieval_query: only when decision == "retrieve". Provide a STANDALONE search string; short (5-15 tokens).
  * BAD Query: "the date" (lacks context).
  * GOOD Query: "graduation ceremony date" (specific).
  * STRATEGY:
    1. Search for the ANCHOR EVENT. (e.g. Question: "What happened 2 days after X?", Query: "timestamp of event X").
    2. Search for the MAPPED ENTITY. (e.g. Question: "Weather in the Windy City", Query: "weather in Chicago").
- detailed_answer: only when decision == "answer"; response using current evidence (keep absolute dates, avoid speculation). If evidence is limited, provide only what is known, or make cautious inferences grounded solely in that limited evidence. Do not mention missing or absent information in this field.
- reasoning: only when decision == "reflect"; if further retrieval is unlikely, use current evidence to think step by step through the evidence and gaps, and work toward the answer, including any time normalization.

Never include extra keys or any text outside the JSON object.

A.2 User prompt of the `generate` node

Apart from the system, the user prompt is responsible for feeding additional information to the LLM. Specifically, at iteration $k+1$, "question" is the original question q. "evidence_block" and "gap_block" are evidence $\mathcal{E}_k$ and gaps $\mathcal{G}_k$ introduced in Sec. 3.2. "raw_block" is the retrieved raw snippets $\mathcal{P}_k$. "reasoning_block" is the reasoning content. "last_query" is the refined query Δq_k introduced in Sec. 3.3 that enables the new query to be different from the prior one. Note that these fields can be left empty if the corresponding information is not present.

USER PROMPT

Question
{question}

Evidence
{evidence_block}

Gaps
{gap_block}

Memory snippets
{raw_block}

Reasoning
{reasoning_block}

Prior Query
{last_query}

INSTRUCTIONS:
1. Update the evidence as a JSON ARRAY of concise factual bullets that directly help answer the question (preserve key numbers/names/time references; use the most specific verified detail without speculation).
2. Update gaps: remove resolved items, add new missing specifics blocking a full answer, and set to "None" when nothing is missing.
3. If you produce a retrieval_query, make sure it differs from the previous query.
4. Decide the next action and return ONLY the JSON object described in the system prompt.

B Formalizing the Markovian Evidence-Gap Tracker

A central component of MemR^3 is the stateful evidence-gap tracker introduced in Sec. 3.2, which formulates the retrieval process as a Markov Decision Process (MDP). This tracker maintains an evolving, compressed epistemic state of i) what information has been reliably established from memory ($\mathcal{E}$) and ii) what specific requirements remain analytically unresolved ($\mathcal{G}$). While the deployable system instantiates this tracker via LLM-generated states, we introduce an idealized formal abstraction below. This abstraction clarifies the intended Markovian transition, enables principled analysis, and provides a theoretical foundation for diagnosing system robustness (e.g., isolating retrieval coverage bias from LLM hallucination bias).

Definition B.1 (Idealized Requirement Space)**.** For a user query q, we define a finite set of *atomic information requirements*, which specify the minimal facts needed to fully answer the query:

$$R(q) = \{r_1, r_2, \ldots, r_m\}. \tag{2}$$

For example, for the question "How many months passed between events A and B?", the requirement set can be

$$R(q) = \{\text{date}(A), \text{date}(B)\}. \tag{3}$$

Each $r \in R(q)$ is associated with a symbolic predicate (e.g., a timestamp, entity attribute, or event relation), and $R(q)$ provides the semantic target against which retrieved memories are judged.

Definition B.2 (Memory-Support Relation)**.** Let $\mathcal{M}$ be the memory store and $\mathcal{P}_k \subseteq \mathcal{M}$ denote the snippets retrieved at iteration k. We define a relation $m \models r$ to indicate that memory item $m \in \mathcal{M}$ contains sufficient information to support requirement $r \in R(q)$. Formally, $m \models r$ holds if the textual content of m contains a minimal witness (e.g., a timestamp, entity mention, or explicit assertion) matching the predicate corresponding to r. The matching criterion may be implemented via deterministic pattern rules or LLM-based semantic matching.

Definition B.3 (Idealized Markovian State Transition)**.** At iteration k, the idealized system maintains an explicit epistemic state $\mathcal{S}_k^\star = (\mathcal{E}_k^\star, \mathcal{G}_k^\star)$. Given newly retrieved snippets $\mathcal{P}_{k-1}$ from the previous routing action, the ideal deterministic state transition is:

$$\mathcal{E}_k^\star = \mathcal{E}_{k-1}^\star \cup \left\{ r \in R(q) \middle| \exists m \in \mathcal{P}_{k-1} \text{ s.t. } m \models r \right\}, \qquad \mathcal{G}_k^\star = R(q) \setminus \mathcal{E}_k^\star. \tag{4}$$

Crucially, this update rule strictly satisfies the Markov property: the new state $\mathcal{S}_k^\star$ depends only on the immediate prior state $\mathcal{S}_{k-1}^\star$ and the newly observed snippets $\mathcal{P}_{k-1}$. The system monotonically accumulates verified requirements and strictly shrinks the gap. Because the state acts as a sufficient statistic, the raw snippets $\mathcal{P}_{k-1}$ can be safely purged once the state is updated without information loss, thereby preventing token usage from growing with the accumulated retrieval history, under the state abstraction adopted in this work.

B.1 Practical Instantiation via LLM as a Stochastic State Estimator

In the deployable MemR^3 system, exact symbolic matching for the memory-support relation ($m \models r$) is intractable for unstructured text. Therefore, the theoretical Markov transition is instantiated

through the LLM, acting as a *stochastic state estimator*:

$$\mathcal{E}_k, \mathcal{G}_k = \texttt{LLM}(q, \mathcal{P}_{k-1}, \mathcal{E}_{k-1}, \mathcal{G}_{k-1}, \text{prompt}), \tag{5}$$

where the prompt explicitly instructs the model to:

(i) extract concise factual evidence from $\mathcal{P}_{k-1}$ to update $\mathcal{E}_{k-1}$,

(ii) re-evaluate and enumerate the missing information to form $\mathcal{G}_k$, and

(iii) strictly avoid speculative inference.

Thus, the empirical state $(\mathcal{E}_k, \mathcal{G}_k)$ serves as a stochastic approximation to the idealized state $(\mathcal{E}_k^\star, \mathcal{G}_k^\star)$:

$$(\mathcal{E}_k, \mathcal{G}_k) \approx f_{\text{trans}}(\mathcal{P}_{k-1}, \mathcal{E}_{k-1}, \mathcal{G}_{k-1}). \tag{6}$$

Deviations from the idealized state arise primarily from LLM extraction noise (e.g., hallucination bias) or semantic granularity mismatches between the text and the latent requirement space. This state-estimation perspective directly informs our diagnostic ablation and error attribution in the main experiments, bridging our rigorous theoretical abstraction with prompt-driven empirical behavior.

B.2 Correctness Properties under Idealized Extraction

Although the practical LLM instantiation lacks deterministic guarantees, the idealized Markovian tracker defined in Definition B.3 satisfies several intuitive properties essential for reliable closed-loop retrieval.

Theorem B.4 (Properties of the Idealized Tracker)**.** *Assume the initialization $\mathcal{E}_0^\star = \varnothing$ (and hence $\mathcal{G}_0^\star = R(q)$). Under the deterministic state transition rule (Definition B.3), the following hold:*

1. ***Monotonicity:*** *$\mathcal{E}_{k-1}^\star \subseteq \mathcal{E}_k^\star$ and $\mathcal{G}_k^\star \subseteq \mathcal{G}_{k-1}^\star$ for all $k \geq 1$.*
2. ***Soundness:*** *If $m \models r$ for some newly retrieved memory $m \in \mathcal{P}_{k-1}$, then $r \in \mathcal{E}_k^\star$.*
3. ***Completeness at convergence:*** *If every requirement $r \in R(q)$ is supported by the accumulated trajectory of retrieved snippets (i.e., $\exists m \in \bigcup_{0 \leq j \leq K-1} \mathcal{P}_j$ such that $m \models r$), then the recursive state updates ensure $\mathcal{E}_K^\star = R(q)$ and hence the gap is closed, $\mathcal{G}_K^\star = \varnothing$.*

Proof. (1) By the Markovian transition rule in Definition B.3,

$$\mathcal{E}_k^\star = \mathcal{E}_{k-1}^\star \cup \big\{ r \in R(q) \,\big|\, \exists m \in \mathcal{P}_{k-1} \text{ s.t. } m \models r \big\}, \tag{7}$$

so it trivially holds that $\mathcal{E}_{k-1}^\star \subseteq \mathcal{E}_k^\star$. Since $\mathcal{G}_k^\star = R(q) \setminus \mathcal{E}_k^\star$ and $\mathcal{E}_{k-1}^\star \subseteq \mathcal{E}_k^\star$, we obtain $\mathcal{G}_k^\star \subseteq \mathcal{G}_{k-1}^\star$.

(2) If $m \models r$ for some $m \in \mathcal{P}_{k-1}$, then by Definition B.3 we have $r \in \{r' \in R(q) \mid \exists m' \in \mathcal{P}_{k-1} \text{ s.t. } m' \models r'\} \subseteq \mathcal{E}_k^\star$.

(3) Let $r \in R(q)$ be arbitrary. By assumption, there exists some retrieved snippet $m \in \bigcup_{0 \leq j \leq K-1} \mathcal{P}_j$ such that $m \models r$. Hence, for some iteration $j \in \{0, \ldots, K-1\}$, we have $m \in \mathcal{P}_j$ and $m \models r$. By the transition rule in Definition B.3, this support is incorporated at the next state update, so $r \in \mathcal{E}_{j+1}^\star$. By the monotonicity property established in Part (1),

$$\mathcal{E}_{j+1}^\star \subseteq \mathcal{E}_{j+2}^\star \subseteq \cdots \subseteq \mathcal{E}_K^\star. \tag{8}$$

Therefore, $r \in \mathcal{E}_K^\star$. Since $r \in R(q)$ was arbitrary, we have $R(q) \subseteq \mathcal{E}_K^\star$. Conversely, $\mathcal{E}_K^\star \subseteq R(q)$ by construction of the update rule. Thus $\mathcal{E}_K^\star = R(q)$, and consequently

$$\mathcal{G}_K^\star = R(q) \setminus \mathcal{E}_K^\star = \emptyset. \tag{9}$$

□

These properties characterize the target Markovian behavior that the stochastic LLM-based tracker aims to approximate, clarifying why a correct state estimate is sufficient to keep context bounded while reasoning converges.

B.3 Approximation Biases and System Limitations

The abstraction in this section assumes access to an ideal tracker that updates $(\mathcal{E}_k^\star, \mathcal{G}_k^\star)$ exactly according to the requirement-support relation $m \models r$. In practice, MemR^3 uses an LLM-generated tracker $(\mathcal{E}_k, \mathcal{G}_k)$, which only approximates this ideal update. This introduces several forms of approximation bias: i) *Coverage bias* (false negatives): supported requirements $r \in R(q)$ that are omitted from $\mathcal{E}_k$; ii) *Hallucination bias* (false positives): requirements r that appear in $\mathcal{E}_k$ even though no retrieved memory item supports them. Additionally, there could be other system limitations like: iii) *Granularity bias*: cases where the tracker records a coarser fact (e.g., a year) but the requirement space $R(q)$ contains a finer predicate (e.g., an exact date), so the ideal requirement is never fully satisfied. We provide a more detailed discussion of these biases in the case studies in Sec. E.8.

C Implementation Details of the Stateful Controller

To instantiate the theoretical Markovian state machine proposed in Section 3.2, we explicitly define a directed graph using the LangGraph framework (LangChain, 2025). The graph consists of several structural nodes (`start`, `end`) and functional nodes (`generate`, `router`, `retrieve`, `reflect`, `answer`). Specifically, `start` is unconditionally followed by `retrieve`, and `end` is reached immediately after `answer`. The `generate` node acts as the stochastic state estimator introduced in Eq. 1. Below, we detail the `router` policy and the three action execution nodes.

Router (The Control Policy). At each iteration k, the `router` acts as an autonomous sequential controller. It evaluates the current epistemic state and selects the next optimal action from $\{\texttt{retrieve}, \texttt{reflect}, \texttt{answer}\}$. Each action a_k is accompanied by a specific textual generation conditioned on the state:

$$a_k \in \{(\texttt{retrieve}, \Delta q_k), (\texttt{reflect}, f_k), (\texttt{answer}, w_k)\},$$

where Δq_k is a targeted refinement query addressing the specific gap, f_k is an internal reasoning trace, and w_k is a draft answer. They are subsequently utilized by the downstream action nodes.

To ensure practical stability and prevent infinite loops in the state machine, the `router` applies three deterministic safety constraints (Algorithm 1): 1) a maximum iteration budget n_{max} that forces termination (`answer`) once exhausted, 2) a reflect-streak capacity n_{cap} that forces exploration (`retrieve`) when too many consecutive internal reflections occur, and 3) a retrieval-opportunity check that switches the action to `reflect` if the observation environment returns no new snippets.

These lightweight rules act as necessary safety rails while preserving operational flexibility. The primary routing signal remains the evolving evidence-gap state, which mathematically determines whether additional retrieval is necessary to close the gap, whether reflection is warranted, and when answering becomes justified.

Retrieve (The Observation Action). When transitioning to the `retrieve` node, the system uses the generated refinement Δq_k to construct $q_k^{\text{ret}} = q \oplus \Delta q_k$ (where $\oplus$ denotes string concatenation), fetching new memory snippets from the environment:

$$\mathcal{P}_k = \texttt{Retrieve}(q_k^{\text{ret}}, \mathcal{M} \setminus \mathcal{M}_{k-1}^{\text{ret}}),\ \mathcal{M}_k^{\text{ret}} = \mathcal{M}_{k-1}^{\text{ret}} \cup \mathcal{P}_k. \tag{10}$$

Crucially, this node implements our "absorb and discard" mechanism. The snippets $\mathcal{P}_k$ are evaluated independently during the next-generation step *without* accumulating the historical raw text $\mathcal{M}_{k-1}^{\text{ret}}$, thereby keeping the prompt footprint bounded by the current state and newest snippets. Furthermore, subtracting $\mathcal{M}_{k-1}^{\text{ret}}$ masks previously retrieved snippets, preventing redundant selection and improving the signal-to-noise ratio per iteration.

A major architectural benefit of MemR^3 is that it treats all concrete retrievers as decoupled observation environments. Any underlying storage, be it vector search, graph memory (e.g., Zep), hybrid stores, or future paradigms, can be seamlessly integrated, provided they return textual snippets and support stable identifiers for masking. This abstraction ensures MemR^3 remains a lightweight, portable, and strictly backend-agnostic controller.

Reflect (Internal State Re-evaluation). Related refinement methods include confidence-guided self-correction (Li et al., 2024) and selective decoding (Tang et al., 2025). The `reflect` node allows the agent to update its epistemic state without querying the external memory store. It incorporates the reasoning trace f_{k-1} and invokes the generator to re-evaluate the state $(\mathcal{E}_k, \mathcal{G}_k)$ via Eq. 1,

enabling the system to deduce implicit conclusions from the existing evidence and dynamically reformulate gaps.

Answer (Terminal Convergence). Once the router determines the gap is closed (or safely terminates via budget constraints), the state machine converges to the `answer` node. The final response is generated using the original query q, the draft answer w_k, and the dense semantic evidence $\mathcal{E}_k$, utilizing the prompt p_w adapted from Rasmussen et al. (2025):

$$w \leftarrow \texttt{LLM}(q, w_k, \mathcal{E}_k, p_w). \tag{11}$$

By grounding the final generation strictly on the filtered evidence $\mathcal{E}_k$ rather than the unbounded raw retrieval context, the LLM is naturally insulated from noisy text. It is explicitly instructed to avoid hallucinations and remain rigorously faithful to the extracted facts.

D Experimental Settings

D.1 Licenses for Datasets

We use two existing conversational memory benchmarks in our experiments: LongMemEval$_s$ (Wu et al., 2025) and LoCoMo (Maharana et al., 2024). LongMemEval$_s$ is released under the MIT License according to its public dataset repository.[1] LoCoMo is released under the Creative Commons Attribution-NonCommercial 4.0 International License (CC-BY-NC-4.0) according to its public dataset repository.[2] We use both datasets only for research evaluation, cite the original dataset papers, and do not redistribute modified versions of the datasets.

D.2 Baselines

As shown in Sec. 4.1, we select four groups of methods: 1) *Full-Memory Reference*: **Full-Context**. 2) *Open-loop Heuristics*: Vanilla **RAG** (Lewis et al., 2020). 3) *Implicit Agentic Retrievers*: **Self-RAG** (Asai et al., 2024) and a custom strong baseline **RCR** (RAG-CoT-RAG) extended from ReAct (Yao et al., 2023). 4) *Memory Systems*: **A-Mem** (Xu et al., 2025), **LangMem** (LangChain Team, 2025), **Mem0** (Chhikara et al., 2025), and graph-based **Zep** (Rasmussen et al., 2025).

D.2.1 Full-Memory Reference

We include **Full-Context** as a strong reference baseline, which provides the model with the entire conversation or memory buffer without retrieval. This setting is unconstrained by retrieval recall, but it can still suffer from context dilution when the relevant evidence is buried in long histories. This setting directly follows Chhikara et al. (2025); Wang & Chen (2025).

D.2.2 Open-loop Heuristics

Vanilla RAG (Lewis et al., 2020). The vanilla RAG retrieves the top-k relevant snippets from the query once and provides a direct answer, without iterative retrieval or reasoning-based refinement. Other retrieval settings (n_{chk}, chunk size, etc.) are the same as those in MemR^3.

D.2.3 Implicit Agentic Retrievers

In this group, we examine the agentic structures underlying memory retrieval to demonstrate MemR^3's advanced performance and, in particular, to highlight the advantage of MemR^3's agentic structure. To validate this, we include Self-RAG (Asai et al., 2024) and design a strong heuristic baseline, RAG-CoT-RAG (RCR), which combines RAG and CoT (Wei et al., 2022).

Self-RAG (Asai et al., 2024). A model-driven retrieval/reflect controller where the LLM decides, at each step, whether to answer or issue a refined retrieval query. Unlike MemR3, retrieval decisions in Self-RAG are implicit in the model's chain-of-thought, without explicit state tracking. We reproduce their original code and prompt to suit our task.

FAIR-RAG (Asl et al., 2025). FAIR-RAG decomposes the question into sub-queries, filters the retrieved evidence, and refines the queries until a structured evidence assessment judges the evidence

[1] `https://huggingface.co/datasets/xiaowu0162/longmemeval`

[2] `https://huggingface.co/datasets/adymaharana/locomo`

sufficient. As no official code is released, we reimplement it from the published prompts with the same chunks, embeddings, and five-iteration budget as MemR^3.

Decomp-RAG. A single LLM call decomposes the question into subqueries, each subquery is retrieved once, and the union of the results is used to answer. The query set is fixed before observing retrieved evidence.

Rewrite-RAG. Each round rewrites the query conditioned on previously retrieved raw documents. This baseline has no persistent evidence-gap state or state-based stopping criterion.

RAG-CoT-RAG (RCR). We design a strong heuristic baseline that extends beyond ReAct (Yao et al., 2023) by performing one initial retrieval (Lewis et al., 2020), a CoT (Wei et al., 2022) step to identify missing information, and a second retrieval using a refined query. It provides multi-step retrieval but lacks an explicit evidence-gap state or a general controller.

D.2.4 Memory Systems

In this group, we consider recent advanced memory systems, including **A-Mem** (Xu et al., 2025), **LangMem** (LangChain Team, 2025), **Mem0** (Chhikara et al., 2025), and Zep (Rasmussen et al., 2025) to demonstrate MemR^3's overall effectiveness from a memory control perspective.

A-Mem (Xu et al., 2025)[3]. A-Mem is an agent memory module that turns interactions into atomic notes and links them into a Zettelkasten-style graph using embeddings plus LLM-based linking.

LangMem (LangChain Team, 2025). LangMem is LangChain's persistent memory layer that extracts key facts from dialogues and stores them in a vector store (e.g., FAISS/Chroma) for later retrieval.

Mem0 (Chhikara et al., 2025)[4]. Mem0 is an open-source memory system that enables an LLM to incrementally summarize, deduplicate, and store factual snippets, with an optional graph-based memory extension.

Zep (Rasmussen et al., 2025). Zep is a hosted memory service that builds a time-aware knowledge graph over conversations and metadata to support fast semantic and temporal queries. We implement their original code and the official Zep service/API.

D.3 Experimental protocols.

The experiments are conducted on a machine with an AMD EPYC 7713P 64-core processor, an A100-SXM4-80GB GPU, and 512GB of RAM. Repeated evaluations use three independent executions under the same configuration.

We first describe how we determine the number of chunks n_{chk} (for RAG backbone) and the number of relevant facts/entities (for Zep backbone) n_{fact} per iteration, as well as the maximum iteration budget n_{max}. Specifically, we choose a nominal configuration for MemR^3 (with a RAG retriever) by arbitrarily setting the number of chunks per iteration $n_{\text{chk}} = 3$ and the max iteration budget $n_{\text{max}} = 5$. In Fig. 5 a), we fix $n_{\text{max}} = 5$ and perform ablations over $n_{\text{chk}} \in \{1, 3, 5, 7, 9\}$. In Fig. 5 b), we also fix $n_{\text{max}} = 5$ and perform ablations over $n_{\text{fact}} \in \{5, 10, 15, 20\}$. In Fig. 5 c), we fix $n_{\text{chk}} = 3$ and perform ablations over $n_{\text{max}} \in \{1, 2, 3, 4, 5\}$. Considering both the LLM-as-a-Judge score and token usage, we eventually choose $n_{\text{chk}} = 5$, $n_{\text{fact}} = 15$, and $n_{\text{max}} = 5$ in all main experiments.

For all chunk-based methods like RAG (Lewis et al., 2020), Self-RAG (Asai et al., 2024), RAG-CoT-RAG, and MemR^3 (RAG retriever), we set the embedding model as `text-embedding-3-large` (OpenAI, 2024a) and use a re-ranking strategy (Reimers & Gurevych, 2019) (`ms-marco-MiniLM-L-12-v2`) to search relevant memories rather than just similar ones. The chunk size is selected from {128, 256, 512, 1024} using GPT-4o-mini backend when $n_{\text{max}} = 1$ and $n_{\text{chk}} = 1$, and we ultimately choose 256, which is also in line with Mem0 (Chhikara et al., 2025).

[3] `https://github.com/WujiangXu/A-mem`

[4] `https://github.com/mem0ai/mem0`

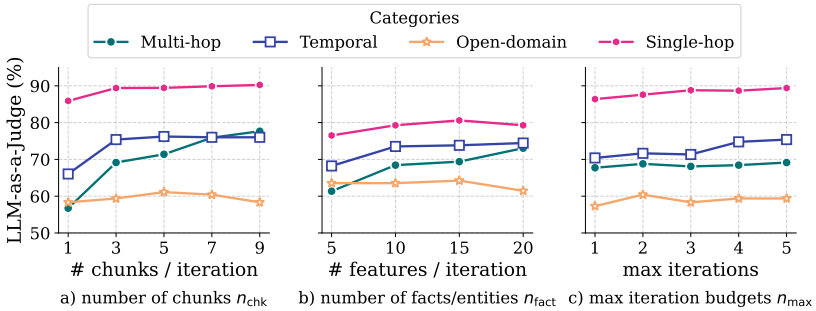

Figure 5: LLM-as-a-Judge score (%) with different a) number of chunks per iteration, b) number of facts/entities, and c) max iterations.

E Experimental Results

E.1 Repeated Experiments

Tables 6-8 report three-run means and sample standard deviations ($n = 3$). Individual runs are listed in Tables 6, 7, and 9. Unless stated otherwise, single-run LongMemEval$_s$ results elsewhere use Run 1, the original evaluation.

Table 6: Three-run J scores (%) on LongMemEval$_s$ with GPT-4o-mini.

	RAG	**Self-RAG**	**RCR**	**Full-Context**	**MemR^3**
Mean ± SD	56.60 ± 0.72	41.53 ± 0.61	56.07 ± 0.31	58.53 ± 1.10	71.67 ± 0.76
Run 1	57.40	42.20	56.00	59.80	72.00
Run 2	56.00	41.00	55.80	57.80	70.80
Run 3	56.40	41.40	56.40	58.00	72.20

Table 7: Three-run comparison with FAIR-RAG on LongMemEval$_s$ across six backends. The last column lists individual runs.

Backend	**Method**	**J (%)**	**Completion tokens**	**Individual runs (J)**
GPT-4o-mini	FAIR-RAG	54.93 ± 0.46	996 ± 4	55.20 / 55.20 / 54.40
	MemR^3	71.67 ± 0.76	357 ± 74	72.00 / 70.80 / 72.20
GPT-4.1-mini	FAIR-RAG	60.00 ± 0.92	$1,060 \pm 29$	60.80 / 60.20 / 59.00
	MemR^3	76.40 ± 1.06	491 ± 5	77.20 / 76.80 / 75.20
GPT-5-mini	FAIR-RAG	73.20 ± 1.40	$10,144 \pm 185$	74.60 / 73.20 / 71.80
	MemR^3	81.80 ± 0.72	$5,029 \pm 78$	82.60 / 81.20 / 81.60
DeepSeek-V3.2	FAIR-RAG	49.60 ± 0.40	$1,931 \pm 11$	49.20 / 49.60 / 50.00
	MemR^3	80.47 ± 1.14	815 ± 136	80.80 / 79.20 / 81.40
Claude-haiku-4.5	FAIR-RAG	66.27 ± 1.17	$3,671 \pm 116$	65.40 / 65.80 / 67.60
	MemR^3	78.80 ± 0.40	$1,019 \pm 3$	78.80 / 78.40 / 79.20
Gemini-2.5-flash	FAIR-RAG	61.47 ± 0.92	$7,327 \pm 1,679$	60.40 / 62.00 / 62.00
	MemR^3	79.13 ± 1.40	460 ± 103	79.00 / 80.60 / 77.80

Table 8: Three-run results on LoCoMo. FAIR-RAG and MemR^3 (RAG) share the same RAG retriever; MemR^3 (Zep) is compared with its Zep backbone.

Method	GPT-4o-mini		GPT-4.1-mini	
	J (%)	**Completion tokens**	**J (%)**	**Completion tokens**
FAIR-RAG	68.96 ± 0.23	891 ± 5	78.46 ± 0.20	786 ± 2
MemR^3 (RAG)	81.55 ± 0.34	300 ± 6	86.75 ± 0.46	431 ± 1
Zep	74.69 ± 0.24	45 ± 2	78.94 ± 0.15	64 ± 2
MemR^3 (Zep)	76.26 ± 0.34	390 ± 1	80.88 ± 0.46	496 ± 4

Table 9: Individual MemR^3 runs on LoCoMo, with category means and sample standard deviations. MH = multi-hop; T = temporal; OD = open-domain; SH = single-hop.

LLM	Retriever	Run	MH	T	OD	SH	Overall
GPT-4o-mini	Zep	1	68.09	73.52	68.75	80.72	76.13
		2	69.86	72.59	67.71	80.36	76.00
		3	70.21	75.39	64.58	80.72	76.65
		Mean ± SD	69.39 ± 1.14	73.83 ± 1.43	67.01 ± 2.17	80.60 ± 0.21	76.26 ± 0.34
	RAG	1	71.63	77.26	61.46	89.28	81.75
		2	70.21	76.01	59.38	89.40	81.16
		3	72.34	75.39	62.50	89.64	81.75
		Mean ± SD	71.39 ± 1.08	76.22 ± 0.95	61.11 ± 1.59	89.44 ± 0.18	81.55 ± 0.34
GPT-4.1-mini	Zep	1	78.72	78.50	72.92	84.34	81.36
		2	75.89	77.26	68.75	84.58	80.44
		3	78.72	77.57	67.71	84.34	80.84
		Mean ± SD	77.78 ± 1.63	77.78 ± 0.65	69.79 ± 2.76	84.42 ± 0.14	80.88 ± 0.46
	RAG	1	81.56	83.18	69.79	91.93	86.79
		2	82.62	80.69	75.00	92.65	87.18
		3	79.43	82.55	69.79	91.93	86.27
		Mean ± SD	81.20 ± 1.62	82.14 ± 1.29	71.53 ± 3.01	92.17 ± 0.42	86.75 ± 0.46

E.2 Category Breakdown on LongMemEval$_s$

Table 10 expands Fig. 3 by category. MemR^3 improves over Full-Context in most settings, particularly single-session preference. With GPT-5-mini, the largest deficits occur in temporal reasoning and multi-session questions, consistent with loss of useful details during compression. Table 12 extends this backend-capability comparison to GPT-5.6 Terra.

E.3 Query Decomposition and Rewriting

MemR^3 outperforms both query-only strategies across the three tested backends (Table 11), supporting the contribution of state-based control beyond query reformulation.

Table 11: J scores (%) for query decomposition and iterative rewriting on LongMemEval$_s$.

Backend	Decomp-RAG	Rewrite-RAG	MemR^3
GPT-4o-mini	62.00	63.40	72.00
GPT-4.1-mini	68.40	68.40	77.20
GPT-5-mini	79.00	76.40	82.60

Table 10: J scores (%) on LongMemEval$_s$. P/A/U = single-session preference/assistant/user; TR = temporal reasoning; MS = multi-session; KU = knowledge update. Improvements are absolute differences from Full-Context. Three-run results are in Tables 6 and 7.

Backend	Method	P	A	TR	MS	KU	U	Overall
GPT-4o-mini	RAG	56.67	94.64	42.11	33.83	70.51	87.14	57.40
	Self-RAG	26.67	83.93	18.05	18.80	66.67	78.57	42.20
	RCR	53.33	92.86	34.59	38.35	71.79	84.29	56.00
	Full-Context	40.00	94.64	39.10	45.11	76.92	88.57	59.80
	MemR^3	60.00	92.86	55.64	67.67	75.64	95.71	72.00
	Improvement	20.00	-1.78	16.54	22.56	-1.28	7.14	12.20
GPT-4.1-mini	RAG	73.33	98.21	45.86	36.84	69.23	90.00	60.80
	Self-RAG	80.00	96.43	32.33	30.08	75.64	94.29	57.20
	RCR	86.67	98.21	63.16	54.89	74.36	92.86	72.20
	Full-Context	56.67	98.21	48.87	50.38	78.21	88.57	65.40
	MemR^3	86.67	100.00	61.65	70.68	76.92	97.14	77.20
	Improvement	30.00	1.79	12.78	20.30	-1.29	8.57	11.80
GPT-5-mini	RAG	86.67	98.21	57.89	38.35	70.51	88.57	65.20
	Self-RAG	63.33	96.43	46.62	53.38	74.36	91.43	65.60
	RCR	83.33	100.00	70.68	62.41	75.64	92.86	76.40
	Full-Context	76.67	98.21	82.71	84.21	85.90	98.57	87.20
	MemR^3	86.67	100.00	69.17	78.20	85.90	97.14	82.60
	Improvement	10.00	1.79	-13.54	-6.01	0.00	-1.43	-4.60
DeepSeek-V3.2	RAG	50.00	94.64	32.33	28.57	57.69	87.14	51.00
	Self-RAG	20.00	58.93	6.77	11.28	29.49	48.57	24.00
	RCR	33.33	62.50	20.30	22.56	50.00	80.00	39.40
	Full-Context	46.67	92.86	35.34	41.35	70.51	90.00	57.20
	MemR^3	86.67	98.21	69.92	72.18	83.33	98.57	80.80
	Improvement	40.00	5.35	34.58	30.83	12.82	8.57	23.60
Claude-haiku-4.5	RAG	70.00	98.21	42.86	36.84	71.79	91.43	60.40
	Self-RAG	56.67	92.86	24.81	28.57	69.23	87.14	51.00
	RCR	53.33	78.57	40.60	48.12	65.38	85.71	57.80
	Full-Context	46.67	82.14	26.32	36.84	70.51	62.86	48.60
	MemR^3	83.33	96.43	66.17	69.92	84.62	97.14	78.80
	Improvement	36.66	14.29	39.85	33.08	14.11	34.28	30.20
Gemini-2.5-flash	RAG	53.33	87.50	42.11	38.35	61.54	80.00	55.20
	Self-RAG	16.67	89.29	12.03	24.06	50.00	75.71	39.00
	RCR	26.67	98.21	33.08	23.31	53.85	90.00	48.60
	Full-Context	40.00	100.00	49.62	61.65	78.21	92.86	68.40
	MemR^3	80.00	98.21	63.91	70.68	89.74	95.71	79.00
	Improvement	40.00	-1.79	14.29	9.03	11.53	2.85	10.60

E.4 Query-Time Cost and Backend Capability

Table 12 reports query-time token counts over all LLM calls. With GPT-4o-mini, MemR^3 uses 5.9% of Full-Context's total tokens.

Backend capability. Full-Context overtakes MemR^3 on GPT-5-mini and GPT-5.6 Terra, while MemR^3 retains its input-token advantage (Table 12).

E.5 Token Efficiency and State Compression on LoCoMo

To empirically validate the structural advantages of our Markovian state machine, we analyze the token footprint versus reasoning accuracy.

Table 12: Average query-time tokens per question on LongMemEval$_s$, including all controller, state-generation, and answer-generation calls. GPT-5-mini completion counts include internal reasoning.

Backend	Method	Input	Completion	Total	J (%)
GPT-4o-mini	RAG	1,403	8	1,411	57.40
	Self-RAG	3,461	159	3,619	42.20
	RCR	4,044	106	4,150	56.00
	MemR^3	6,195	314	6,509	72.00
	Full-Context	109,624	12	109,636	59.80
GPT-4.1-mini	MemR^3	6,596	486	7,082	77.20
	Full-Context	109,624	11	109,634	65.40
GPT-5-mini	MemR^3	6,886	5,065	11,951	82.60
	Full-Context	109,624	447	110,071	87.20
GPT-5.6 Terra	MemR^3	6,428	415	6,843	85.80
	Full-Context	109,624	46	109,670	88.80

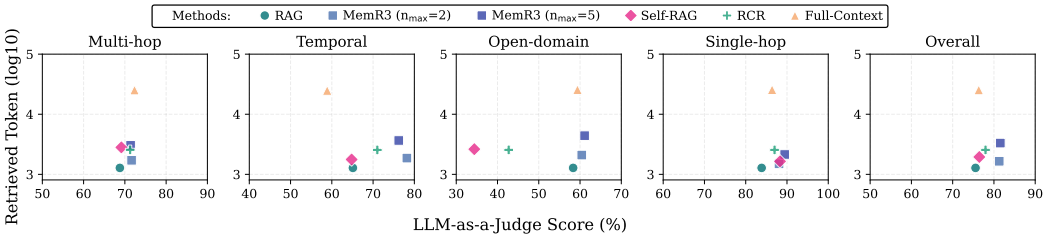

Figure 6: Average token usage of the retrieved context (y-axis) and LLM-as-a-Judge (J) Score (x-axis) of RAG, Self-RAG, RCR, MemR^3, and Full-Context across four categories on LoCoMo.

Unlike implicit agentic retrievers (Self-RAG, RCR) that append newly retrieved chunks to the context window, MemR^3 compresses observations into an explicit state and masks assimilated raw text. As illustrated in Figure 6, MemR^3 occupies the high-accuracy, low-token region, improving the trade-off that constrains traditional autoregressive accumulators. On LoCoMo, $n_{\max} = 2$ uses fewer tokens and offers a favorable accuracy-cost trade-off, but exceeds $n_{\max} = 5$ in accuracy only for open-domain questions; the larger budget performs better in the other three categories.

E.6 Lexical Metrics

Table 13 reports EM and F1 alongside the LLM-as-a-Judge score on LongMemEval$_s$. The default MemR^3 does not constrain answer length and often produces explanatory answers, which overlap little with the short references. MemR^3 (concise) reuses the same retrieval trajectory, evidence state, and draft answer, and only changes the final instruction to request an answer of at most six words. This raises EM/F1 from 0.00/17.72 to 19.00/43.49, while J decreases only from 72.00 to 68.40, indicating that the low lexical scores of the default setting largely reflect output format.

Table 13: Auxiliary metrics (%) on LongMemEval$_s$ with GPT-4o-mini.

Metric	RAG	Self-RAG	RCR	Full-Context	MemR^3 (default)	MemR^3 (concise)
EM	23.60	0.20	0.00	24.20	0.00	19.00
F1	43.21	14.28	13.90	46.48	17.72	43.49
J	57.40	42.20	56.00	59.80	72.00	68.40

E.7 Convergence and Stopping

We inspect the iteration frequency of MemR^3 in Figure 7. Across both benchmarks, the distribution exhibits a clear bimodal pattern that is consistent with our evidence-gap controller. On **LoCoMo**, most queries terminate after either 1 iteration or at the maximum budget, with relatively few cases

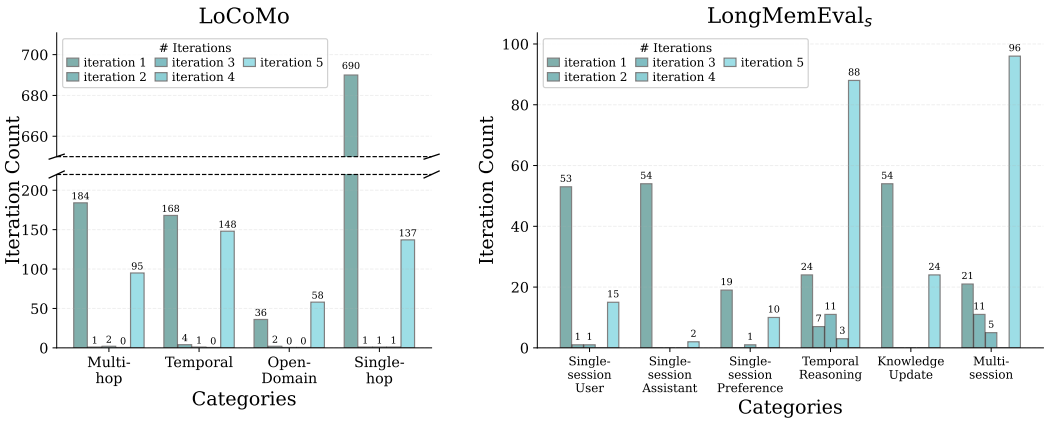

Figure 7: Number of questions requiring different numbers of iterations before final answers, across four categories on LoCoMo (left) and six categories on LongMemEval$_s$ (right).

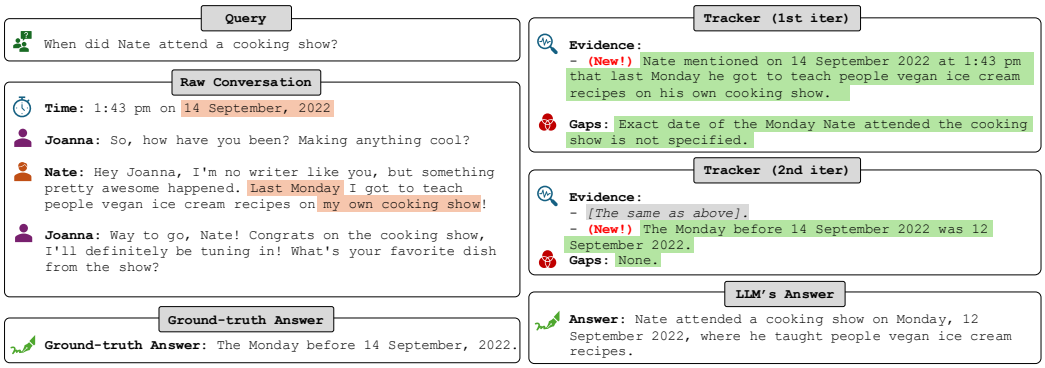

Figure 8: A successful case executed by MemR^3 on LoCoMo.

stopping in the middle. In particular, *Single-hop* questions are overwhelmingly resolved in a single step, while *Multi-hop*, *Temporal*, and *Open-Domain* questions show a larger fraction of budget-exhausted runs, reflecting cases where the gap cannot be fully closed from the retrieved evidence. On **LongMemEval**$_s$, the same pattern persists across all six categories: *Single-session User*, *Single-session Assistant*, *Single-session Preference*, and *Knowledge Update* are dominated by 1-iteration termination, whereas *Temporal Reasoning* and especially *Multi-session* contain many 5-iteration cases, indicating that these categories more often require repeated retrieval or remain unresolved under the available evidence.

Stopping policies. Stopping after two or three unchanged states reduces accuracy (Table 14); a temporary stall can still be followed by useful retrieval. This trajectory-dependent rule differs from assigning a common iteration budget to all questions, as in Fig. 6.

Table 14: J scores (%) on LongMemEval$_s$ with GPT-4o-mini. Stall-m stops after m consecutive iterations without evidence-state updates.

Method	RAG	Full-Context	MemR^3 (stall-2)	MemR^3 (stall-3)	MemR^3
J (%)	57.40	59.80	66.00	67.40	72.00

E.8 Case Studies.

Successful case. Fig. 8 shows a successful case. From the perspective of our idealized requirement space $R(q)$ in Appendix B, this example corresponds to the case where the tracker is both sound and (approximately) complete. The requirement set for the query "*When did Nate attend a cooking show?*" can be abstracted into two atomic requirements: the anchor timestamp of Nate's message

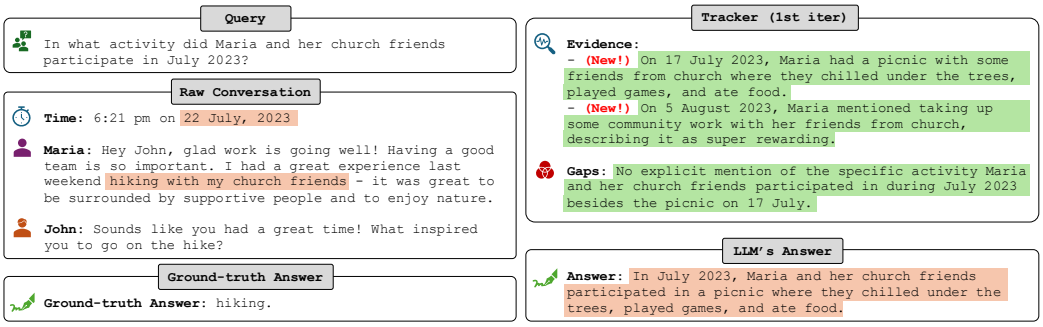

Figure 9: A failed case executed by MemR^3 with coverage bias (false negatives). Although the retrieved snippets contain evidence supporting the requirement, the LLM-based tracker fails to extract and record it in $\mathcal{E}_k$.

on 14 September 2022 and the mapping from "*last Monday*" to an absolute calendar date. After the first iteration, the tracker correctly records only the anchor requirement as evidence and keeps the exact date as a remaining gap, consistent with the monotonic update rule where evidence grows while gaps shrink. Once the second snippet explicitly states that "*The Monday before 14 September 2022 was 12 September 2022*", the date requirement becomes supported, and the gap set becomes empty, illustrating the completeness property of the ideal tracker and showing that MemR^3 can closely approximate this behavior in practice.

Next, we also present three types of failed (or structurally non-convergent) cases: coverage bias, hallucination bias, and granularity bias.

Failed case with coverage bias (false negatives). Fig. 9 shows a failed case with coverage bias. Here, the requirement space $R(q)$ for the query "*In what activity did Maria and her church friends participate in July 2023?*" contains an atomic requirement r_{act} corresponding to the activity "*hiking*". The raw memory snippet explicitly supports r_{act}, so under the ideal update rule, we would have $m \vDash r_{\text{act}}$ for some retrieved memory item $m \in \mathcal{P}_{k-1}$, and thus $r_{\text{act}} \in \mathcal{E}_k^\star$ once that snippet is retrieved. However, in this specific iteration, the tracker failed to extract the "*hiking*" activity due to interference from other activities. Consequently, the practical tracker could not include this requirement in the evidence set and instead focused on the retrieved picnic-related facts, so $r_{\text{act}} \notin \mathcal{E}_k$. This is exactly a coverage bias (false negative): a supported requirement in $R(q)$ is omitted from the evidence, leading to an incorrect answer.

Failed case with hallucination bias (false positives). Fig. 10 shows a failed case with hallucination bias, which illustrates the complementary failure mode where the tracker adds unsupported content to the evidence. For the query "*When did Andrew go rock climbing?*", the requirement space $R(q)$ includes a date requirement r_{date} for the rock-climbing event. The retrieved snippet only supports a weaker predicate stating that the event happened on "*last Sunday*" relative to 13 June 2023, i.e., it provides a relative temporal relation but not a fully grounded calendar date. Under the idealized semantics, no memory item satisfies $m \vDash r_{\text{date}}(\text{2023-06-13})$; thus, r_{date} should remain in the gap set until additional information is retrieved. In contrast, the practical tracker directly records "*Andrew went rock climbing on 13 June, 2023*" as evidence and sets the gap to "*None*", introducing a requirement in $\mathcal{E}_k$ that is not supported by any memory item. This is precisely a hallucination bias (a false positive), and it explains why the final answer overspecifies the date to 13 June rather than the correct 11 June.

Case with granularity bias. In Fig. 11, for the question "*When did Melanie paint a sunrise?*", the correct answer is simply "*2022*" (the year). Under the ideal abstraction, however, the requirement space $R(q)$ implicitly contains a fine-grained predicate r_{date} corresponding to the full year-month-day of the painting event. The memory store only contains a coarse statement such as "*Melanie painted the lake sunrise image last year (2022).*"

In the ideal tracker, no memory item m satisfies $m \models r_{\text{date}}$, so the precondition of Theorem B.4's completeness clause is violated and the ideal gap $\mathcal{G}_k$ never becomes empty. The practical LLM tracker mirrors this behavior: it quickly recovers the year 2022 as evidence, but continues to treat

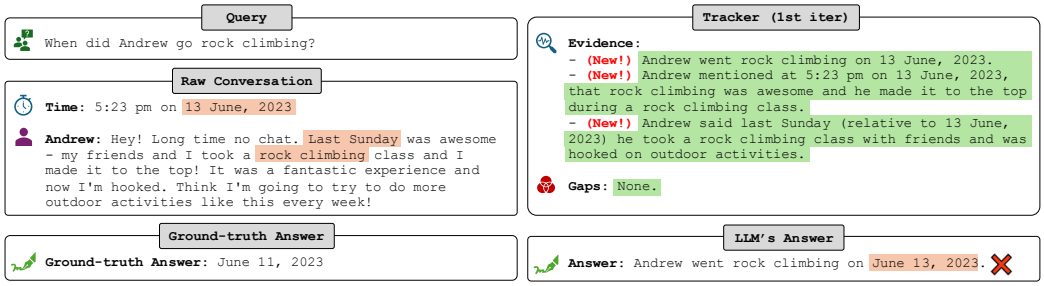

Figure 10: A failed case executed by MemR^3 with hallucination bias (false positives).

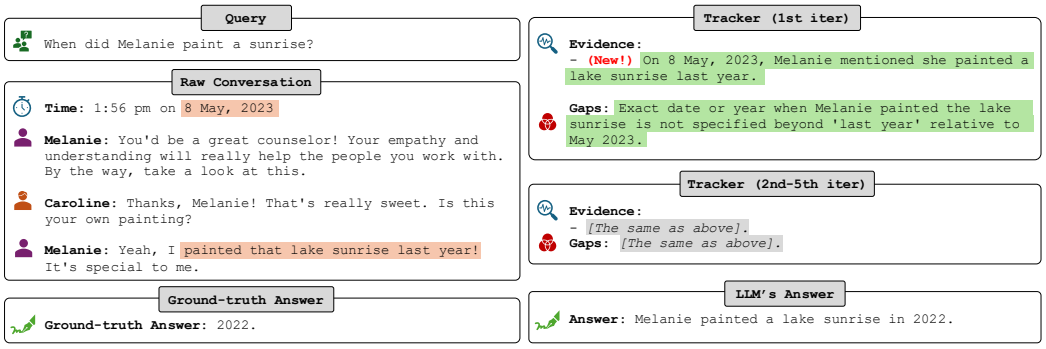

Figure 11: A case executed by MemR^3 with granularity bias. Although the final answer is correct, it cannot close the loop until the max iteration is reached.

the exact date as a remaining gap, eventually hitting the iteration budget without fully closing $\mathcal{G}_k$. This example shows that some apparent "failures" of the approximate tracker are in fact structural: they arise from a mismatch between the granularity of $R(q)$ and the information actually present in the memory store.

Based on the above case studies, we suggest that future studies focus on relieving the aforementioned biases (coverage, hallucination, granularity) to keep memory systems accurate and efficient.

F Re-alignment of LoCoMo dataset

Misalignment in existing works. Although the correct order of the different categories is not explicitly reported in LoCoMo (Maharana et al., 2024), we can infer it from the number of questions in each category. The correct alignment is shown in Table 15. We believe this clarification could benefit the LLM memory community.

Table 15: The alignment of the category orders and names in the LoCoMo dataset.

Category	Category 1	Category 2	Category 3	Category 4	Category 5
Name	Multi-Hop	Temporal	Open-Domain	Single-Hop	Adversarial
# Questions	282	321	96	830	445

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Yes. The abstract and introduction accurately summarize the paper's proposed evidence-gap controller, its scope in conversational memory, and the empirical efficiency-accuracy claims supported by the experiments.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Yes. The paper explicitly discusses limitations including dependence on backend recall, LLM state-estimation errors, limited large-scale backend comparisons, and restriction to text-only memory.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Yes. The theoretical claims are stated under an explicit idealized abstraction, with assumptions and formal proofs provided in the appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Yes. The paper provides the datasets, baselines, metrics, prompts, model backends, and key hyperparameters needed to reproduce the main experimental results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Yes. An anonymized code link is released.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Yes. The paper specifies the datasets, baselines, evaluation metrics, model backends, and main controller settings used in the experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Yes. The paper reports the averaged runs and standard deviations on LoCoMo datasets.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Yes. The computer resources are shown in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: To the best of our knowledge, the research complies with the NeurIPS Code of Ethics and does not involve human-subject experimentation or other ethically sensitive procedures.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper briefly discusses the broader impacts in the conclusion.

    Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a new high-risk model or scraped dataset that would require special release safeguards.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Yes. The paper cites the external datasets and systems used and explicitly documents the licenses.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not introduce or release a new dataset, model, or code asset as part of the submission.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or experiments with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human-subject research requiring IRB approval or an equivalent review.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: The paper clearly states that LLMs are a core component of the method, serving as the state estimator, router, and answer generator.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.