RETRIEVAL-DRIVEN MEMORY RECONSOLIDATION FOR LONG-TERM LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-term memory is essential for LLM-based agents operating over extended interactions. Many existing systems primarily update memory when new information arrives, treating retrieval as the endpoint of memory access rather than a driver of memory evolution. Recent methods have begun to exploit signals from memory use, but it remains less explored how retrieved evidence can guide agent-selected edits to a typed memory structure. Inspired by memory reconsolidation in cognitive neuroscience, we propose **REALM**, a **r**econsolidation-**e**volution **a**gentic **l**ong-term **m**emory framework. REALM organizes experiences into a heterogeneous cognitive graph, retrieves evidence through adaptively composed graph-search atoms, and performs retrieval-driven reconsolidation by using the activated subgraph and interaction feedback to add, strengthen, or weaken typed relations. Under our evaluation protocol, REALM achieves average accuracies of 75.97% on LoCoMo and 65.11% on LongMemEval, corresponding to gains of 7.17 and 1.90 points over the strongest reported baseline averages. Ablation studies show that memory reconsolidation improves overall performance on both benchmarks, while further analyses indicate that it progressively reorganizes related memory units into more coherent local structures for collective evidence recall and utilization during reasoning. These results support retrieval-driven reconsolidation as one mechanism for adaptive long-term memory in LLM agents.

1 INTRODUCTION

Recent advances in Large Language Model (LLM)-based agents have enabled their deployment in long-term interactive scenarios like personal assistants, long-horizon task collaboration, and continual decision-making (Zhang et al., 2025; Du, 2026). In these scenarios, agents must accumulate knowledge across extensive histories and leverage past experiences for future reasoning, making long-term memory a foundational capability of intelligent agents (Wang et al., 2024; Li et al., 2026; Fang et al., 2026).

As shown in Figure 1, existing long-term memory research predominantly follows a forward evolution paradigm, where memory updates, including addition, deletion, and merging of memory units, are primarily triggered by newly acquired information (Zhong et al., 2024; Hu et al., 2025; Wang & Chen, 2025). Recent studies have begun to exploit signals from memory use. RMM uses citations produced during response generation to adapt a learned reranker, whereas HeLa-Mem updates episodic association weights from retrieval-time co-activation and subsequently distills highly connected clusters into semantic memory (Tan et al., 2025; Zhu et al., 2026a). These studies demonstrate that memory utilization can influence future access and organization. However, they leave open how an agent can use retrieved evidence and feedback from a particular task to select typed structural edits within the activated memory subgraph.

This open problem is closely related to memory reconsolidation in cognitive neuroscience, a foundational mechanism in which recalled memory is not static data (Adam, 2026; Lee & Jung, 2025; Samieiyeganeh et al., 2026). When a memory is reactivated, it can enter a labile state in which its relational structure may be strengthened, weakened, or newly formed, allowing memory networks to actively evolve through use and enabling them to remain adaptable to new experiences while maintaining stability (Nader et al., 2000a). This perspective suggests that long-term memory should be modeled as a dynamic lifecycle in which memory organization can be revisited after retrieval rather

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

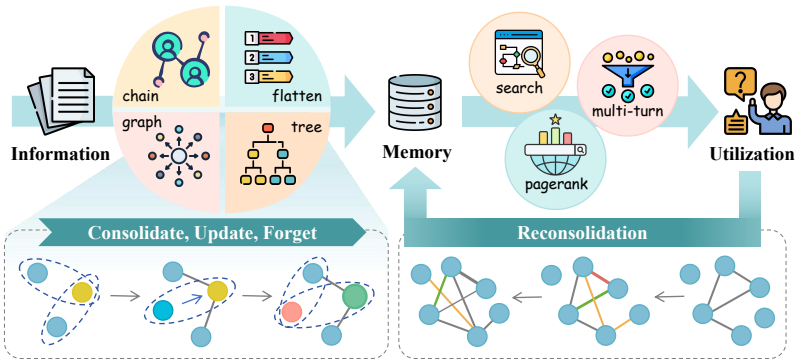

Figure 1: **Motivation of retrieval-driven memory reconsolidation.** Many memory systems still update memory primarily from newly acquired information, while relying on predefined structures or fixed retrieval procedures for organization and access. REALM forms a closed-loop lifecycle, leveraging retrieval feedback to dynamically reorganize the memory structure upon utilization.

than fixed after construction. Realizing this form of post-retrieval evolution in an agent requires both a memory structure that supports local relational edits and a retrieval mechanism that can operate over its changing topology (Hu et al., 2026).

To address this challenge, we propose **REALM**, **a reconsolidation-evolution agentic long-term memory lifecycle framework**, which instantiates this holistic lifecycle through three complementary components: (1) Autonomous Organization: We define atomic memory nodes and relations, empowering agents to autonomously construct local topologies that organically emerge into a unified agentic cognitive graph. (2) Adaptive Retrieval: We decompose complex graph traversal into atomic search actions, allowing agents to compose context-tailored retrieval policies dynamically. (3) Memory Reconsolidation: After each interaction, we perform local reconsolidation on the activated subgraph, dynamically updating connections to support continual retrieval-driven evolution. Experiments on standard benchmarks demonstrate that REALM consistently outperforms conventional post-static memory paradigms, with further analyses confirming that memory reconsolidation effectively optimizes memory structures over long horizons. At the same time, the agentic cognitive graph enables efficient memory organization and retrieval. These findings support our hypothesis that long-term memory should evolve continuously through the agent's memory use, becoming an integral part of its evolving intelligence.

Our main contributions are summarized as follows:

- **Retrieval-Driven Memory Reconsolidation.** We formulate memory evolution as a reconsolidation step, where an LLM agent uses the retrieved subgraph and task feedback to operate over typed relations, including adding, strengthening, and weakening edges.
- **REALM Framework.** We propose a unified agentic memory framework that integrates autonomous memory organization, adaptive retrieval, and post-retrieval reconsolidation into an evolving memory lifecycle.
- **Empirical Validation.** Comprehensive evaluations show that REALM consistently improves long-term memory performance, revealing the autonomous emergence of reusable memory topologies and retrieval policies.

2 METHOD

2.1 PROBLEM FORMULATION

We model a long-horizon agent interacting with an environment over an information stream $\mathcal{O} = \{o_1, o_2, \cdots, o_T\}$. Conventional agent memory paradigms sequentially evolve a memory graph $\mathcal{G}_t$ upon receiving o_t via forward operations including insertion, consolidation, or forgetting, and passively extract a static subgraph $\mathcal{G}_q \subset \mathcal{G}_t$ to answer a query q. In contrast, we contend that memory utilization yields crucial cognitive feedback that should actively drive structural refinement. We therefore introduce a post-retrieval memory reconsolidation phase. Given a query q and its answer

feedback f, the agent dynamically reorganizes the activated topology $\mathcal{G}_q$ into an optimized state $\mathcal{G}'_q$. The global memory state transition is thus formulated as:

$$\mathcal{G}_{t+1} = \text{Reconsolidating}((\mathcal{G}_t \setminus \mathcal{G}_q) \cup \mathcal{G}'_q,\ f). \tag{1}$$

The overall framework is illustrated in Figure 2.

2.2 Memory Organization: Unified Cognitive Graph

2.2.1 Graph Representation

To enable schema-free local configurations, we model memory as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where nodes correspond to different categories of memory units and edges encode their relations. Decomposing the structure into logically orthogonal dimensions, we define the node-type space as $\mathcal{K}_V = \{\text{entity}, \text{event}, \text{episode}, \text{fact}\}$, and the edge-type space as $\mathcal{K}_E = \{\text{logical}, \text{causal}, \text{hierarchical}, \text{associative}\}$. Each agent-generated cognitive node $v \in \mathcal{V}$ is structured as:

$$v = \big(\kappa_v,\ c_v,\ des_v,\ k_v,\ t_v\big), \qquad \kappa_v \in \mathcal{K}_V, \tag{2}$$

where κ_v denotes the node type, c_v is the raw source content, des_v represents the semantic description, k_v signifies the type-specific keywords or names, and t_v specifies the temporal span constraint if applicable. Correspondingly, each agent-generated directed edge $e_{ij} \in \mathcal{E}$ is formalized as:

$$e_{ij} = \big(v_i,\ v_j,\ \kappa_{ij},\ des_{ij},\ w_{ij}\big), \qquad \kappa_{ij} \in \mathcal{K}_E, \tag{3}$$

where κ_{ij} represents the relation type, des_{ij} is the relational description text, and $w_{ij} \in [0, 1]$ denotes the confidence weight. This formulation decouples raw memory content from topology, providing the essential structural variables for downstream retrieval and topology evolution.

2.2.2 Memory Construction

Unlike conventional methods that separate insertion, merging, and forgetting, REALM unifies these operations into a single integration phase, treating forgetting as redundant (Ong et al., 2025). Upon receiving new observation o_t, the agent first extracts m candidate memory units $X = \{x_i\}_{i=1}^{m}$ with the context of the current conversation summary s_t and the recent information c_t, where each $x_i \in \mathcal{V}$ is autonomously typed by the agent. Then for each candidate unit $x \in X$, the agent queries its related neighborhood nodes: $N(x) = N_{sim}(x) \cup N_{recent}(x)$, where N_{sim} and N_{recent} denote semantically similar and recently processed nodes, respectively. Conditioned on $N(x)$, the agent determines whether a new candidate should be *added* to memory as a new node, *merged* with an existing memory node, or *skipped* and not saved in memory. If add or merge is selected, the agent subsequently infers a set of directed relational edges E_v between x and $N(x)$ within the predefined edge space $\mathcal{E}$. This unified pipeline empowers the agent to autonomously determine not only *whether to remember* an observation through selective retention, but also *how to represent* it via dynamic topological integration.

2.3 Memory Retrieval: Strategy Atom Combination

Building on existing retrieval methods (Peng et al., 2025), REALM formalizes graph search into a dynamic pipeline of three sequential phases: seeding, expanding, and filtering. Rather than employing rigid heuristics, the agent autonomously formulates a composite retrieval policy π:

$$\pi = \pi_{seed} \oplus \pi_{expand} \oplus \pi_{filter} \tag{4}$$

based on the real-time search state, where each phase-specific policy combines atomic strategies from a predefined action space:

$$\pi_p = (a_1^p, a_2^p, \cdots, a_k^p), \quad a_i^p \in \mathcal{A}_p. \tag{5}$$

2.3.1 Seed Localization

Given a query q, the agent first generates π_{seed} consisting of multiple parameterized retrieval plans to identify starting nodes via keyword, query, or hybrid matching modes. Each plan is formalized as a tuple $p_i = (Q_i, K_i, T_i, \tau_i)$, explicitly defining the query string Q_i, keywords K_i, target node types T_i, and temporal constraints τ_i. The aggregate initial seed set is formed via $\{S_i\} = \{\text{Retrieve}_{\text{seed}}(\mathcal{G}, p_i)\}$.

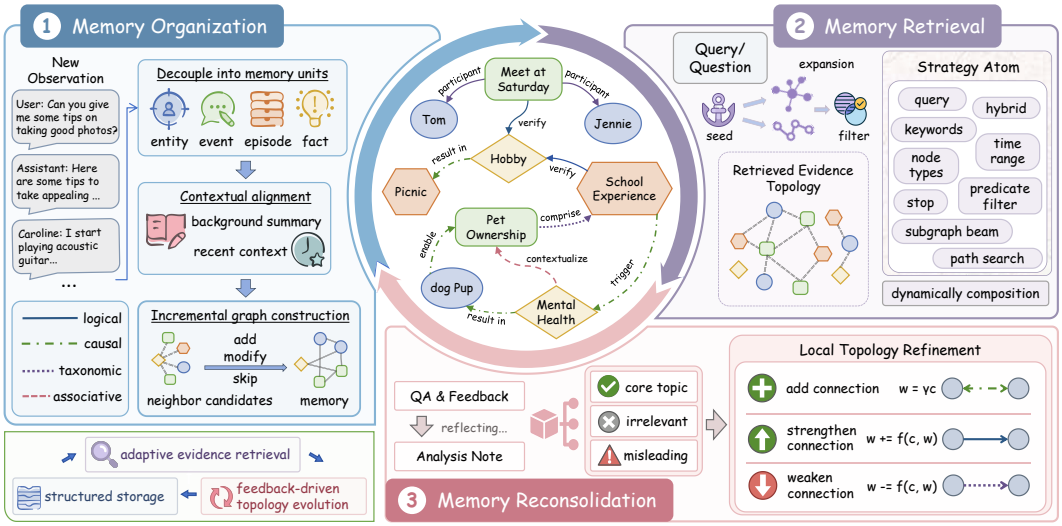

Figure 2: **Overview of REALM.** The framework organizes observations into an agentic cognitive graph memory, retrieves evidence through adaptive graph-search atoms, and reconsolidates the activated subgraph to evolve memory topology after use.

2.3.2 Adaptive Evidence Expansion

Starting from the seeds, the agent iteratively expands the evidence graph. At iteration t, it evaluates the collected memories R_t (where $R_0 = \bigcup S_i$) to determine whether R_t already provides enough evidence to answer the query. If sufficiency is met, search terminates; otherwise, it selects an active frontier $F_t \subseteq R_t$. For each frontier node $v \in F_t$, the LLM predicts an individualized expansion action:

$$a_v = (\text{mode},\ \text{predicate},\ \text{decay},\ \text{inhibit}), \tag{6}$$

where components regulate the traversal behavior, edge-type predicates, temporal decay, and path inhibition, respectively. The newly discovered nodes are aggregated as

$$N_t = \bigcup_{v \in F_t} \text{Expand}(v, a_v), \tag{7}$$

which update the state via $R_{t+1} = R_t \cup N_t$. During graph exploration, each newly discovered node $v_i \in N_t$ reached via edge e_{ij} receives a context-aware access score $s_{\text{aces}}(v)$, formulated as:

$$s_{\text{aces}}(v) = \beta s_{\text{sim}}(q,\ des_{ij}) + (1-\beta) w_{ij} - \Delta_{decay}(t_v), \tag{8}$$

where s_{sim} measures the semantic similarity between query q and edge description des_{ij}, w_{ij} is the topological relation confidence, $\Delta_{decay}(t_v)$ scales the temporal penalty based on the permitted expansion decay and node's temporal span t_v, and β is a balancing coefficient. Detailed strategy atoms and hyper-parameters are provided in Appendix E.1. The expansion loop repeats until topological convergence or the predefined search depth and memory budgets are exhausted.

2.3.3 Evidence Aggregation

Upon termination, the retrieved set R is globally reranked. The final score for each node combines its accumulation weight and semantic relevance:

$$\text{Score}(v) = \alpha s_{\text{aces}}(v) + (1-\alpha) s_{\text{sim}}(v, q). \tag{9}$$

The top-K nodes constitute the final evidence set $\mathcal{V}_q = \text{Top}_K(R)$ for downstream generation. Crucially, $\mathcal{V}_q$ induces the activated local cognitive subgraph $\mathcal{G}_q = (\mathcal{V}_q, \mathcal{E}_q)$, where $\mathcal{E}_q = \{(u,v) \in \mathcal{E} \mid u, v \in \mathcal{V}_q\}$, which serves as the direct topology input for the subsequent memory reconsolidation.

2.4 Memory Reconsolidation: Topology Evolution

While memory organization structures the graph and retrieval utilizes it, reconsolidation leverages retrieval feedback to iteratively evolve the memory topology $\mathcal{G}$ beyond its initial semantic constraints into a usage-aware cognitive structure.

2.4.1 RETRIEVAL-INDUCED STRUCTURAL FEEDBACK

For each interaction, the retrieval phase yields the activated subgraph $\mathcal{G}_q = (\mathcal{V}_q, \mathcal{E}_q)$ and task feedback $f = (\mathrm{ans}, \hat{\mathrm{ans}}, y)$, where ans is the agent-generated answer, $\hat{\mathrm{ans}}$ is the ground-truth answer, and $y \in \{0, 1\}$ indicates whether the agent's answer is correct. Such outcome feedback is available in practical settings where task results are observable and verified. Rather than modifying memory content, REALM exploits the topological co-utilization patterns within $\mathcal{V}_q$. The agent first distills contextual guidance by inferring the underlying topic structure:

$$T_q = \textsc{InferTopicStructure}(q, \mathcal{V}_q), \tag{10}$$

which summarizes the semantic relations among recalled memories to guide subsequent refinement.

2.4.2 LOCAL TOPOLOGY RECONSOLIDATION

We model reconsolidation as an agent-driven topology decision process. Conditioned on $\mathcal{G}_q$, f and T_q, the agent generates a modification set $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$, where each decision is parameterized as a 5-tuple $d = (v_i, v_j, a, r, c)$. Here, $v_i, v_j \in \mathcal{V}_q$ denote target nodes, r is the relation type, $c \in [0, 1]$ is the decision confidence, and $a \in \{\text{add}, \text{strengthen}, \text{weaken}\}$ defines the atomic edge operation space. The agent may create new relations between previously disconnected memories when they are repeatedly activated under the same reasoning context. Existing relations can be strengthened when they consistently support successful retrieval, while weak or misleading relations can be suppressed when they repeatedly introduce irrelevant evidence.

Confidence-Guided Topology Update. To execute modifications without exceeding the valid edge weight boundaries, we unify the adaptive bounded adjustment. The confidence score determines the magnitude of topology modification. Let w_{ij} be the current edge weight and $w_{\min}$ be the minimum allowable threshold. We first define a bounded representation $w_b = \max(w_{\min}, \min(1, w_{ij}))$. The weight updating pipeline is then compactly formulated as:

$$w_{ij} = \begin{cases} \gamma \cdot c, & a = \text{add}, \\ \leftarrow \text{Clip}\Big(w_{ij} + \gamma \cdot c,\ w_{\min},\ 1\Big), & a \in \{\text{strengthen,weaken}\}, \end{cases} \tag{11}$$

where

$$\gamma = \begin{cases} 1 - w_b, & a = \text{strengthen}, \\ -(w_b - w_{\min}), & a = \text{weaken}. \end{cases} \tag{12}$$

This formulation smoothly regulates the update scale near the weight boundaries, protecting the topology from destabilization by isolated or duplicate retrieval instances.

2.4.3 CLOSED-LOOP MEMORY EVOLUTION

As formalized in Eq. (1), this mechanism establishes a seamless, closed-loop memory lifecycle:

$$\mathcal{G}_t \xrightarrow{\text{Retrieve}} (\mathcal{G}_q, f) \xrightarrow{\text{Reconsolidate}} \mathcal{G}_{t+1}. \tag{13}$$

Through continuous task-driven interaction, the memory graph progressively adapts from an initially semantic organization into a usage-aware cognitive structure. Frequently co-utilized memory paths are dynamically reinforced while ineffective retrieval pathways are gradually weakened. Therefore, long-term memory is no longer a static storage component, but an evolving cognitive graph.

3 EXPERIMENT

To comprehensively evaluate the proposed framework, we conduct experiments from four complementary perspectives. Specifically, we aim to answer the following research questions:

RQ1. Does REALM consistently outperform existing long-term memory systems on representative long-term memory benchmarks?

RQ2. Does memory reconsolidation contribute to performance improvement, and is its effectiveness robust under different interaction orders?

Table 1: Accuracy (%) on LoCoMo by question category using GPT-4o-mini as both backbone and LLM judge model. **Bold** and underlined values indicate the best and second-best results respectively.

Method	LoCoMo-Category				Average
	Multi Hop	Temporal	Open Domain	Single Hop	
MIRIX (Wang & Chen, 2025)	54.26	68.54	46.88	68.22	64.33
Mem0 (Chhikara et al., 2025)	58.75	52.34	45.83	73.33	64.57
Zep (Rasmussen et al., 2025)	52.12	54.82	33.33	66.23	59.22
MAGMA (Jiang et al., 2026a)	52.80	65.00	51.70	77.60	68.80
Nemori (Nan et al., 2025)	56.90	64.90	48.50	76.40	68.70
A-Mem (Xu et al., 2026)	53.55	50.16	41.67	61.83	56.62
REALM (Ours)	**64.54**	**76.64**	**58.33**	**81.57**	**75.97**

Table 2: Accuracy (%) on LongMemEval using GPT-4o-mini as both backbone and LLM judge model. **Bold** and underlined values indicate the best and second-best results, respectively.

Method	LongMemEval-Category						Average
	single-session preference	single-session assistant	temporal reasoning	multi-session	knowledge update	single-session user	
MIRIX	53.30	63.60	25.60	30.10	52.60	72.90	43.49
Zep	53.30	75.00	54.10	47.40	74.40	**92.90**	63.21
MAGMA	**73.30**	**83.90**	45.10	**50.40**	66.70	72.90	59.98
Nemori	62.70	73.20	43.00	51.40	52.60	77.70	56.20
REALM (Ours)	36.66	82.14	**56.69**	46.28	**88.89**	89.06	**65.11**

RQ3. What enables REALM to remain effective prior to reconsolidation, and how does agentic autonomy shape memory organization and retrieval?

RQ4. How does memory reconsolidation continuously reshape memory organization and influence future retrieval behaviors?

3.1 Experimental Setup

Benchmarks & Baselines. We evaluate REALM on two standard long-term memory benchmarks: LoCoMo (Maharana et al., 2024) and LongMemEval_S (Wu et al., 2024). We compare REALM against representative systems across three principal paradigms: (1) flat and temporal retrieval (Mem0, Zep); (2) multi-component and agentic adaptive memory (MIRIX, A-Mem, Nemori); and (3) strategy-guided graph architectures (MAGMA).

Metrics. Following existing works, we employ LLM-as-a-Judge (Zheng et al., 2023) for accuracy evaluation, excluding unanswerable queries. To ensure strict consistency, all baseline results from both newly conducted experiments and aligned literature utilize identical judge prompts and models.

Detailed descriptions are deferred to Appendix B.

3.2 RQ1: Overall Performance

Table 1 and Table 2 summarize the overall performance of REALM compared with representative long-term memory systems. The evaluation protocol and question scheduling strategy are provided in Appendix B.4. Our method obtains the highest reported overall average on both benchmarks, demonstrating the effectiveness of the proposed framework in diverse long-term memory scenarios. On LoCoMo, REALM achieves an average score of 75.97, outperforming the strongest baseline by 7.17 points and ranking first across all four question categories. On LongMemEval, REALM also obtains the highest overall average (65.11), surpassing the previous best method by 1.90 points. The results indicate that REALM performs robustly across different long-term memory benchmarks.

Looking into different question categories, the performance gains are particularly evident on memory-intensive reasoning tasks. On LoCoMo, REALM improves multi-hop and temporal reasoning by more than 5 and 8 points, respectively, while also consistently outperforming existing methods on open domain and single-hop questions. These results indicate that the proposed framework is effective across both complex reasoning tasks requiring long-range memory integration and

Table 3: Effect of reconsolidation across different categories on LoCoMo and LongMemEval. Subscripts report absolute gains from enabling reconsolidation.

Category	LoCoMo		LongMemEval	
	w/o recon	w/ recon	w/o recon	w/ recon
Multiple	59.57	64.54$_{+4.97}$	43.80	46.28$_{+2.48}$
Temporal	74.14	76.64$_{+2.50}$	55.12	56.69$_{+1.57}$
Single(-Preference)	81.09	81.57$_{+0.48}$	30.00	36.66$_{+6.66}$
Open/Update	53.12	58.33$_{+5.21}$	84.72	88.89$_{+4.17}$
Average	73.96	75.97$_{+2.01}$	62.98	65.11$_{+2.13}$

more straightforward retrieval scenarios. A similar trend is observed on LongMemEval. REALM achieves the best performance on temporal reasoning and knowledge update, while remaining competitive on single-session assistant and single-session user. In particular, the improvement on knowledge update suggests that the framework effectively handles continuously evolving user information. In contrast, the gains on temporal reasoning demonstrate its ability to leverage long-term interaction histories. Other analysis is provided in Appendix C. Overall, the results demonstrate that REALM provides a more effective long-term memory framework than existing approaches across diverse benchmarks and task types.

3.3 RQ2: Effect of Memory Reconsolidation

Memory reconsolidation is the central adaptive component of REALM. To evaluate its effectiveness, we conduct two ablation studies: (1)removing the reconsolidation module while keeping all other components unchanged, and (2)randomizing the question order on LoCoMo to test whether the gains depend on a particular interaction sequence.

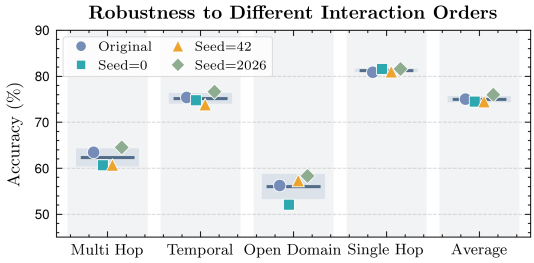

Figure 3: **Robustness to randomized question orders on LoCoMo.** Results remain stable across original and shuffled orders, suggesting that reconsolidation improves memory topology rather than overfitting to a particular query sequence. The shaded region denotes performance variation.

Memory Reconsolidation Effects. Table 3 shows that removing memory reconsolidation consistently degrades performance on both benchmarks. On LoCoMo, enabling reconsolidation improves the overall accuracy by 2.01 points, with the largest gains observed on multi-hop (+4.97) and open-domain (+5.21) questions. Similar improvements are observed on LongMemEval, where reconsolidation increases the overall performance by 2.13 points, particularly benefiting single-session preference (+6.66) and knowledge update (+4.17). These improvements indicate that updating the cognitive graph after each retrieval enables the system to progressively strengthen useful memory associations and improve subsequent retrieval quality.

Robustness to Interaction Order. One possible concern is that the gains from reconsolidation may arise from adapting to a specific question sequence. To verify this, we evaluate REALM under multiple randomly shuffled question orders on LoCoMo. As shown in Figure 3, the performance across different random seeds remains highly consistent for all question categories, with only minor fluctuations around the mean. This result demonstrates that the effectiveness of memory reconsolidation is largely independent of the interaction order. Rather than memorizing a particular sequence of queries, reconsolidation continuously refines the memory graph according to the retrieval process itself, leading to stable improvements under different evaluation orders.

3.4 RQ3: Dissecting Agentic Memory Autonomy

This section investigates why REALM remains competitive even without reconsolidation by analyzing memory organization, utilization, and retrieval strategy selection.

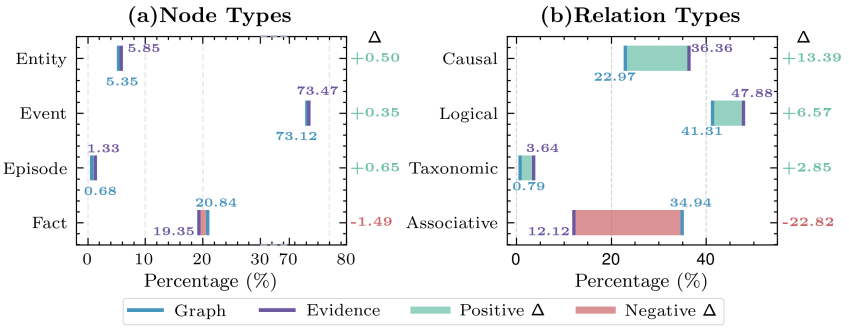

Figure 4: **Deviation between graph composition and evidence retrieval.** (a) Heterogeneous memory node types remain unbiased in both graph structure and evidence utilization. (b) Different relationship types are selectively exploited according to downstream reasoning requirements.

Table 4: Effect of expansion strategy selection on questions requiring graph expansion. Adaptive selection slightly improves over fixed path search and subgraph beam, showing the value and current limitation of strategy composition.

Strategy	**Acc (%)**	**Correct (#)**
path_search only	39.84	147
subgraph_beam only	43.09	159
adaptive selection	**43.36**	**160**

Memory Organization and Utilization. We first compare the distributions of node and relation types in the cognitive graph with those in the retrieved evidence, as shown in Figure 4. Node distributions are highly consistent, with differences below 1.5 percentage points across all types. This suggests that each memory type contributes to retrieval according to both its semantic role and its frequency in the graph. Even less frequent types, such as episode nodes, are retrieved when needed rather than overlooked, indicating that the automatically constructed graph preserves diverse memory forms without introducing retrieval bias. Relation usage exhibits a much stronger task-dependent pattern. During retrieval, causal and logical relations increase by 13.39 and 6.57 percentage points, respectively, whereas associative relations become markedly less frequent. This suggests that different relation types naturally play distinct functional roles: associative relations mainly maintain broad semantic connectivity, while causal and logical relations form the primary reasoning paths for question answering. Overall, these results show that the heterogeneous cognitive graph is not only automatically organized by the agent but also effectively exploited to support downstream reasoning.

Retrieval Efficiency. We further analyze the expansion depth at which supporting evidence is first retrieved. As shown in Figure 5, 84.87% of LoCoMo evidence is found immediately from the seed nodes, found directly from the seed nodes, with only a small fraction requiring one to three additional expansion steps. On LongMemEval, 40.52% of evidence is retrieved directly from the seed layer, whereas a considerable proportion is found only after reaching the maximum expansion depth. This pattern reflects two complementary properties of the proposed retrieval mechanism. For relatively straightforward queries, relevant memories are organized close to effective retrieval entry points, enabling efficient retrieval with minimal graph exploration. For more challenging questions, the cognitive graph still provides sufficient structural connectivity for the retrieval agent to progressively discover distant evidence through multi-hop expansion, indicating both retrieval efficiency and scalability.

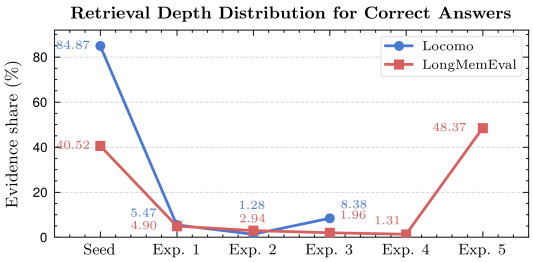

Figure 5: **Retrieval depth for correct answers on LoCoMo and LongMemEval.** The proportion first decreases and then increases with expansion depth, indicating that straightforward queries are answered early, and more challenging ones require deeper graph exploration.

Adaptive Retrieval Strategy. Finally, we evaluate whether allowing the agent to compose retrieval strategies adaptively is beneficial. Table 4 shows that adaptive strategy selection achieves the highest accuracy, slightly outperforming either fixed expansion strategy. Although the improvement over always using "subgraph_beam" is modest, this result is consistent with the observed strategy distribution, where the agent selects subgraph-based expansion for most queries while switching to alternative traversal strategies only when necessary. This behavior suggests that adaptive composition does not seek diversity for its own sake; instead, it learns to recover the strongest default strategy while preserving the flexibility to handle structurally different retrieval scenarios. At the same time, the relatively small margin also reveals a limitation of the current design: the benefit of adaptive retrieval ultimately depends on reliable strategy selection, and inaccurate decisions may reduce the advantage over carefully designed fixed strategies. These analyses suggest that agents can also improve by autonomously constructing memory that aligns well with retrieval demands.

3.5 RQ4: Memory Evolution through Reconsolidation

This section investigates how memory reconsolidation continuously improves future retrieval through graph evolution. We compare retrieval behaviors before and after topology evolution to understand how reconsolidation influences evidence discovery and utilization over continual interactions.

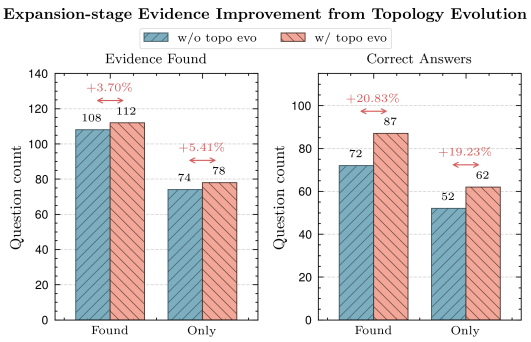

Figure 6: **Evidence gains from topology evolution during the expansion stage.** "Found" denotes cases where the supporting evidence is retrieved during expansion, and "Only" those where it is retrieved exclusively during expansion.

As shown in Figure 6, after enabling memory reconsolidation, more questions successfully retrieve supporting evidence during graph expansion, resulting in a corresponding increase in correctly answered questions. More importantly, topology evolution brings substantially larger improvements in evidence utilization than in evidence discovery. The proportion of questions that successfully retrieve supporting evidence increases by only 3.70% and 5.41%, whereas the proportion of correctly answered questions after evidence retrieval increases by 20.83% and 19.23%, respectively. This indicates that memory reconsolidation does not primarily improve performance by retrieving substantially more memories. Instead, it continuously reorganizes the cognitive graph so that retrieved evidence becomes more relevant and informative for downstream reasoning.

These results suggest that the value of memory reconsolidation lies not in expanding the retrievable memory space, but in continuously improving the quality of memory organization through retrieval-driven graph evolution. As interactions accumulate, the cognitive graph becomes increasingly aligned with future retrieval requirements, complementing static memory organization.

4 Conclusion

This work studies long-term memory in LLM agents from a retrieval-driven lifecycle perspective. Rather than treating retrieval as the terminal step of memory use, REALM uses activated evidence and interaction outcomes to drive subsequent memory reconsolidation. It organizes experiences in a heterogeneous cognitive graph and enables an LLM agent to select constrained operations over typed relations in the retrieved subgraph, including adding, strengthening, and weakening connections. Experiments on LoCoMo and LongMemEval show that reconsolidation improves performance over the corresponding system with a static memory topology. Further analyses indicate that topology evolution organizes scattered interactions into compact structural subgraphs that facilitate collective recall. These findings support a broader view of long-term agent memory as a system shaped by both information acquisition and memory use. We hope that shifting long-term memory from a flat accumulation model toward a retrieval-driven lifecycle will inspire future research into adaptive memory foundations that continuously adapt their organization as agents accumulate experience.

REFERENCES

Abuelgasim Mohamed Ibrahim Adam. Eval-driven memory (edm): A persistence governance layer for reliable agentic ai via metric-guided selective consolidation. *Preprints*, January 2026. doi: 10.20944/preprints202601.0195.v1. URL https://doi.org/10.20944/preprints202601.0195.v1.

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

Pengfei Du. Memory for autonomous llm agents: Mechanisms, evaluation, and emerging frontiers. *arXiv preprint arXiv:2603.07670*, 2026.

Xingbo Du, Loka Li, Duzhen Zhang, and Le Song. Memr^3: Memory retrieval via reflective reasoning for llm agents. *arXiv preprint arXiv:2512.20237*, 2025.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, et al. Lightmem: Lightweight and efficient memory-augmented generation. *arXiv preprint arXiv:2510.18866*, 2025.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17490–17502, 2026.

Joaquín Gonzalez, Mihály Vöröslakos, Deren Aykan, Nina Soto, Noam Nitzan, Rachel Swanson, Mursel Karadas, Zhe Sage Chen, and György Buzsáki. Subspace communication in the hippocampal–retrosplenial axis. *Nature*, pp. 1–10, 2026.

Yingjie Gu, Wenjian Xiong, Liqiang Wang, Pengcheng Ren, Chao Li, Xiaojing Zhang, Yijuan Guo, Qi Sun, Jingyao Ma, and Shidang Shi. Fsfm: A biologically-inspired framework for selective forgetting of agent memory. *arXiv preprint arXiv:2604.20300*, 2026.

B. J. Gutiérrez, Y. Shu, Y. Gu, M. Yasunaga, and Y. Su. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in neural information processing systems*, 37: 59532–59569, 2024.

Sen Hu, Yuxiang Wei, Jiaxin Ran, Xueran Han, Zhiyuan Yao, Huacan Wang, Ronghao Chen, and Lei Zou. Does memory need graphs? a unified framework and empirical analysis for long-term dialog memory. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 26758–26782, 2026.

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

D. Jiang, Y. Li, G. Li, and B. Li. Magma: A multi-graph based agentic memory architecture for ai agents. *arXiv preprint arXiv:2601.03236*, 2026a.

Hanqi Jiang, Junhao Chen, Yi Pan, Ling Chen, Weihang You, Yifan Zhou, Ruidong Zhang, Yohannes Abate, and Tianming Liu. Synapse: Empowering llm agents with episodic-semantic memory via spreading activation. *arXiv preprint arXiv:2601.02744*, 2026b.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory os of ai agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 25972–25981, 2025.

LangChain Inc. Langmem documentation. https://langchain-ai.github.io/langmem/, 2025. Accessed: 2026-06-04.

Jong Won Lee and Min Whan Jung. Memory consolidation from a reinforcement learning perspective. *Frontiers in Computational Neuroscience*, 18:1538741, 2025.

Rui Li, Zeyu Zhang, Xiaohe Bo, Zihang Tian, Xu Chen, Quanyu Dai, Zhenhua Dong, and Ruiming Tang. Cam: A constructivist view of agentic memory for llm-based reading comprehension. *Advances in Neural Information Processing Systems*, 38:113381–113406, 2026.

Z. Li, C. Xi, C. Li, D. Chen, B. Chen, S. Song, S. Niu, H. Wang, J. Yang, C. Tang, et al. Memos: A memory os for ai system. *arXiv preprint arXiv:2507.03724*, 2025.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13851–13870, 2024.

Karim Nader, Glenn E Schafe, and Joseph E Le Doux. Fear memories require protein synthesis in the amygdala for reconsolidation after retrieval. *Nature*, 406(6797):722–726, 2000a.

Karim Nader, Glenn E Schafe, and Joseph E LeDoux. The labile nature of consolidation theory. *Nature reviews neuroscience*, 1(3):216–219, 2000b.

Jiayan Nan, Wenquan Ma, Wenlong Wu, and Yize Chen. Nemori: Self-organizing agent memory inspired by cognitive science. *arXiv preprint arXiv:2508.03341*, 2025.

Kai Tzu-iunn Ong, Namyoung Kim, Minju Gwak, Hyungjoo Chae, Taeyoon Kwon, Yohan Jo, Seung-won Hwang, Dongha Lee, and Jinyoung Yeo. Towards lifelong dialogue agents via timeline-based memory management. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8631–8661, 2025.

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

Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. Graph retrieval-augmented generation: A survey. *ACM Transactions on Information Systems*, 44(2):1–52, 2025.

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

A. Rezazadeh, Z. Li, W. Wei, and Y. Bao. From isolated conversations to hierarchical schemas: Dynamic tree memory representation for llms. In *International Conference on Learning Representations*, volume 2025, pp. 990–1023, 2025.

Mehdi Samieiyeganeh, Parisa Bahraminikoo, Soobia Saeed, Saraswathy Gunasekaran, and Saadat Ahmed. Forgetting as control: A theoretical framework for selective behavioral erasure in post-deployment reinforcement learning agents. *Preprints*, June 2026. doi: 10.20944/preprints202606.0911.v1. URL https://doi.org/10.20944/preprints202606.0911.v1.

Zhen Tan, Jun Yan, I-Hung Hsu, Rujun Han, Zifeng Wang, Long Le, Yiwen Song, Yanfei Chen, Hamid Palangi, George Lee, et al. In prospect and retrospect: Reflective memory management for long-term personalized dialogue agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8416–8439, 2025.

Kaixiang Wang, Yidan Lin, Jiong Lou, Zhaojiacheng Zhou, Bunyod Suvonov, and Jie Li. E-mem: Multi-agent based episodic context reconstruction for llm agent memory. *arXiv preprint arXiv:2601.21714*, 2026.

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

Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li, Xian Li, Bing Yin, et al. Memoryllm: Towards self-updatable large language models. *arXiv preprint arXiv:2402.04624*, 2024.

Yu Wang, Dmitry Krotov, Yuanzhe Hu, Yifan Gao, Wangchunshu Zhou, Julian McAuley, Dan Gutfreund, Rogerio Feris, and Zexue He. M+: Extending memoryllm with scalable long-term memory. *arXiv preprint arXiv:2502.00592*, 2025.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. *arXiv preprint arXiv:2410.10813*, 2024.

W. Xu, Z. Liang, K. Mei, H. Gao, J. Tan, and Y. Zhang. A-mem: Agentic memory for llm agents. *Advances in Neural Information Processing Systems*, 38:17577–17604, 2026.

B. Yan, C. Li, H. Qian, S. Lu, and Z. Liu. General agentic memory via deep research. *arXiv preprint arXiv:2511.18423*, 2025.

Ke Yang, Zixi Chen, Xuan He, Jize Jiang, Michel Galley, Chenglong Wang, Jianfeng Gao, Jiawei Han, and ChengXiang Zhai. Plugmem: A task-agnostic plugin memory module for llm agents. *arXiv preprint arXiv:2603.03296*, 2026.

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

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

Z. Zhang, Q. Dai, X. Bo, C. Ma, R. Li, X. Chen, J. Zhu, Z. Dong, and J.-R. Wen. A survey on the memory mechanism of large language model-based agents. *ACM Transactions on Information Systems*, 43(6):1–47, 2025.

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

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

Jinchang Zhu, Jindong Li, Cheng Zhang, Jiahong Liu, and Menglin Yang. Hela-mem: Hebbian learning and associative memory for llm agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13757–13769, 2026a.

Qiming Zhu, Shunian Chen, Rui Yu, Zhehao Wu, and Benyou Wang. From lossy to verified: A provenance-aware tiered memory for agents. *arXiv preprint arXiv:2602.17913*, 2026b.

A Related Work

Agent Long-term Memory. Existing long-term memory systems mainly focus on memory organization and retrieval, continuously accumulating, organizing, and utilizing historical information (LangChain Inc., 2025). Early works rely on flat incremental stores to extract and summarize historical interactions (Zhong et al., 2024; Chhikara et al., 2025; Wang et al., 2025). To enhance representation, recent approaches adopt explicit structures, including hierarchical trees or mind maps (Rezazadeh et al., 2025; Li et al., 2026; Xu et al., 2026), knowledge graphs (Edge et al., 2024; Yang et al., 2026), temporal (Rasmussen et al., 2025) or OS-inspired management (Li et al., 2025; Packer et al., 2023; Kang et al., 2025; Zhu et al., 2026b), and multimodal representations (Wang & Chen, 2025). Correspondingly, retrieval research enhances utilization via multi-hop graph reasoning (Gutiérrez et al., 2024), iterative search (Yan et al., 2025; Du et al., 2025), dual-process retrieval (You et al., 2026) and adaptive graph traversal Jiang et al. (2026a). However, these methods typically follow predefined organizations and keep memory states static after retrieval. In contrast, we formulate a unified memory lifecycle that extends organization and retrieval with dynamic, autonomous memory reorganization.

Agent Memory Evolution and Lifecycle. Inspired by cognitive neuroscience, incorporating memory stages like encoding and consolidation has gained traction (Gutiérrez et al., 2024; Wang et al., 2026; Jiang et al., 2026b). Neuroscience reveals that recalled memories become transiently labile and undergo reconsolidation for continual adaptation (Nader et al., 2000b), implying that long-term memory evolves through repeated use rather than one-time storage (Gonzalez et al., 2026). While several studies explore memory evolution via offline compression (Fang et al., 2025), recency and frequency heuristics (Zhong et al., 2024; LangChain Inc., 2025), or temporal decay with selective forgetting (Gu et al., 2026), their updates are driven by time or new inputs rather than by memory usage itself. Another approach uses reinforcement learning to estimate memory values (Zhang et al., 2026), but it optimizes isolated entries without modifying the relational structure. RMM uses citations in generated responses to update a learned reranker, while its prospective reflection organizes dialogue into topic-based memory entries (Tan et al., 2025). HeLa-Mem updates episodic association weights from retrieval-time co-activation through a Hebbian rule, and separately distills highly connected clusters into semantic memory with a reflective agent (Zhu et al., 2026a). These studies establish important forms of usage-driven adaptation. REALM instead uses the retrieved subgraph and task outcome to let an agent select constrained add, strengthen, or weaken operations over typed relations, coupling future graph access to the evidence of a particular topic.

B Experimental Details

B.1 Benchmarks

We evaluated REALM on two commonly used long-term memory benchmarks.

- **LoCoMo (Maharana et al., 2024).** LoCoMo evaluates LLM-based agents over extended interaction histories. The dataset contains 10 samples, each consisting of a long multi-session conversation with multiple turns per session, accompanied by a large set of questions requiring agents to locate, integrate, and reason over information distributed across the conversation history. Excluding unanswerable questions, the dataset contains 1,540 questions, including 841 single-hop, 282 multi-hop, 321 temporal, and 96 open-domain questions. Each sample is thus associated with multiple sessions and a large number of questions, ranging from dozens to hundreds, providing a persistent interaction history for evaluating long-term memory.
- **LongMemEval (Wu et al., 2024).** LongMemEval focuses on the long-context and long-term memory abilities of LLM-based agents. We evaluate on LongMemEval_S, which contains samples consisting of multi-session, multi-turn interaction histories, with each sample associated with a single question. Excluding unanswerable questions, the dataset contains 470 questions across six types: 30 single-session preference, 56 single-session assistant, 127 temporal reasoning, 121 multi-session, 72 knowledge-update, and 64 single-session user questions. Unlike LoCoMo, where each conversation sample is paired with multiple questions, each LongMemEval sample is designed to evaluate the agent's ability to retrieve and reason over the relevant information from a long interaction history for one individual query. However, this single-question-per-sample design fails to accommodate our formulation of continuously evolving memory via retrieval feedback. We therefore adapt LongMemEval accordingly as described in Appendix B.4.3.

Following prior agent memory studies (Li et al., 2025; Nan et al., 2025), we exclude unanswerable questions from both benchmarks. We choose LoCoMo and LongMemEval_S because they are the two most widely used benchmarks for evaluating long-term memory in LLM-based agents. Our goal is not to optimize leaderboard performance, but to validate the effectiveness of memory reconsolidation as a continual memory organization mechanism.

B.2 Baselines

We compare our method against representative long-term memory systems covering the major design paradigms in agent memory. Mem0 (Chhikara et al., 2025) represents the classical paradigm of memory extraction and management with flat retrieval. MIRIX (Wang & Chen, 2025) decomposes memory into multiple components, while Zep (Rasmussen et al., 2025) introduces temporal memory structures. A-Mem (Xu et al., 2026) dynamically determines memory formation through agentic decisions. Nemori (Nan et al., 2025) further models memory evolution through adaptive

memory granularity and predictive retrieval. MAGMA (Jiang et al., 2026a) integrates a multi-graph memory architecture with strategy-guided graph traversal to enhance long-term memory. Together, these methods cover the principal design paradigms of modern long-term agent memory systems, providing a comprehensive set of baselines for comparison. The following is a detailed description of each baseline method.

- **Mem0 (Chhikara et al., 2025).** Mem0 maintains a flat memory store through an extraction-update pipeline: an LLM first extracts salient facts from each new message pair conditioned on a running conversation summary and recent history, then a second LLM call retrieves the top-s semantically similar existing memories and issues one of four tool-call operations (`ADD`, `UPDATE`, `DELETE`, `NOOP`) to reconcile the new fact with the existing store, without modeling relations between memories.
- **MIRIX (Wang & Chen, 2025).** MIRIX decomposes memory into six predefined components (Core, Episodic, Semantic, Procedural, Resource, and Knowledge Vault), each maintained by a dedicated memory manager agent coordinated by a meta memory manager that routes incoming information to the relevant components. At inference time, an active retrieval mechanism first infers a topic from the ongoing context and then retrieves the top-k entries from each memory component independently.
- **Zep (Rasmussen et al., 2025).** Zep organizes memory as a temporally-aware knowledge graph (Graphiti) comprising three hierarchical subgraphs: episodes (raw messages), semantic entities and facts extracted from episodes, and higher-level community summaries obtained via label propagation. Facts carry bi-temporal validity intervals, and newly ingested edges can invalidate contradicting existing edges. Retrieval combines cosine similarity, BM25, and breadth-first graph search, followed by reranking.
- **A-Mem (Xu et al., 2026).** Following the Zettelkasten method, A-Mem represents each interaction as an atomic note enriched with LLM-generated keywords, tags, and a contextual description. When a new note is added, the system retrieves its top-k nearest notes by embedding similarity and prompts an LLM to decide which links to establish among them; the same neighborhood is then passed to a separate memory evolution step, where an LLM updates each neighbor's context, keywords, and tags in light of the new note, allowing existing memories to evolve as new experience arrives.
- **Nemori (Nan et al., 2025).** Nemori separates memory construction into episodic integration and semantic distillation. Raw messages are first partitioned into coherent episodes and rewritten into narrative form; each episode is then compared against an LLM-synthesized anticipatory schema retrieved from existing knowledge, and only the prediction-error residual, i.e., information the schema fails to anticipate, is distilled into semantic memory, following the predictive-coding principle that predictable content is redundant.
- **MAGMA (Jiang et al., 2026a).** MAGMA represents each memory event across four orthogonal relation graphs (semantic, temporal, causal, and entity). Retrieval is formulated as policy-guided graph traversal: a router classifies query intent (`WHY`/`WHEN`/`ENTITY`) to reweight edge types, fuses anchor nodes across embedding, keyword, and temporal signals via reciprocal rank fusion, and performs an intent-weighted beam search before linearizing the retrieved subgraph into context. Memory evolution follows a dual-stream design that decouples fast temporal-edge ingestion from asynchronous LLM-based causal/entity edge consolidation.

B.3 Memory Graph Configuration

During retrieval, the number of seed plans for LoCoMo is limited to 1, with a maximum expansion depth of 3; for LongMemEval, the number of seed plans is limited to 3, with a maximum expansion depth of 5; and the maximum number of retrieved nodes is 10. The α value for evidence aggregation is set to 0.8, and the β value for access score is set to 0.6.After each question is evaluated, memory reconsolidation is performed once, and the memory graph is updated. Each question is answered only once, simulating the continuous evolution of memory in real-world usage scenarios. Our framework does not require specialized hardware and can be executed on standard computing environments. All experiments rely on API-based LLM inference and lightweight graph operations.

B.4 Evaluation Setup and Protocol

B.4.1 Online Evaluation Protocol

We evaluate REALM under an online memory evolution protocol, in which memory is updated only after the agent completes each question. For LoCoMo (Maharana et al., 2024), each sample contains a long conversation and multiple questions of different types. We therefore evaluate these questions sequentially, invoking one round of memory reconsolidation after each question. The feedback obtained from the current interaction is used only for the subsequent reconsolidation step and can affect the memory state for later questions.

Each question is evaluated only once. We do not revisit previously evaluated questions, perform multiple training epochs over the evaluation set, or optimize the memory against future questions. Therefore, although feedback from the current interaction can influence subsequent memory organization, no information from a question is used to modify memory before that question is evaluated. This protocol prevents test-question information from being retrospectively incorporated into memory before its initial evaluation.

B.4.2 Feedback Definition and Availability

Under this online protocol, the outcome of each completed interaction provides the feedback used for reconsolidation. After the agent produces ans, an LLM-based judge compares it with the reference answer $\hat{\mathrm{ans}}$ and produces a correctness signal $y \in 0, 1$. Thus, the feedback associated with an interaction is

$$f = (\mathrm{ans}, \hat{\mathrm{ans}}, y). \tag{14}$$

The reconsolidation process receives this feedback together with the activated subgraph, as specified in Algorithm 4 and the corresponding reconsolidation prompt in Prompt F.7.

This setting captures an interactive scenario in which response outcomes can be verified and fed back to the agent. In tasks with objectively verifiable answers, an agent may receive feedback on whether its response is correct and, when available, the correct answer itself. The feedback is therefore an outcome signal from the completed interaction rather than additional training supervision introduced specifically for reconsolidation.

B.4.3 LongMemEval Question Augmentation

The online protocol described above cannot be directly applied to the original LongMemEval (Wu et al., 2024) samples because each sample contains only one benchmark question. Evaluating this question directly would provide at most one reconsolidation step, leaving no subsequent interaction on which to evaluate the resulting memory evolution. We therefore construct a small number of preceding interactions from the oracle sessions while keeping the original LongMemEval question as the final evaluation target.

For each sample, we generate up to six additional questions following the taxonomy of LongMemEval: "single-session-user", "single-session-assistant", "single-session-preference", "multi-session", "temporal-reasoning", and "knowledge-update". The candidate question types are determined according to the number of available sessions: the three single-session types are considered for all samples, while the three multi-session types are considered only when at least two sessions are available. An LLM then determines which candidate types are applicable to the available sessions. For samples with multiple sessions, "multi-session" is retained as a candidate even if it is not selected by the applicability judgment, ensuring that multi-session historical interactions are represented.

GPT-4o-mini is employed for applicability assessment as well as the generation and verification of all question types except multi-session ones, while GPT-4.1 is used exclusively for multi-session questions. For each selected type, the corresponding model generates a question-answer pair from the oracle sessions and verifies whether the generated answer is supported by the available sessions. Only verified questions are retained. The complete procedure is summarized in Algorithm 1. We provide several representative prompts in Prompt B.1; the complete set of prompts will be released together with the source code upon public release.

Algorithm 1 LongMemEval Question Augmentation

Require: Oracle sessions $\mathcal{S}$, session dates $\mathcal{D}$, target question date d_q
Ensure: Verified synthetic question set $\mathcal{Q}_{\text{syn}}$

$\mathcal{Q}_{\text{syn}} \leftarrow \emptyset$
Define question types:
  $\mathcal{T}_{\text{single}} \leftarrow \{\text{single-session-user}, \text{single-session-assistant}, \text{single-session-preference}\}$
  $\mathcal{T}_{\text{multi}} \leftarrow \{\text{multi-session}, \text{temporal-reasoning}, \text{knowledge-update}\}$
$\mathcal{T}_{\text{cand}} \leftarrow \mathcal{T}_{\text{single}}$
if $|\mathcal{S}| \geq 2$ **then**
  $\mathcal{T}_{\text{cand}} \leftarrow \mathcal{T}_{\text{cand}} \cup \mathcal{T}_{\text{multi}}$
end if
Use gpt-4o-mini to determine applicable types $\mathcal{T}_{\text{app}} \subseteq \mathcal{T}_{\text{cand}}$ based on the available sessions
if $|\mathcal{S}| \geq 2$ **then**
  $\mathcal{T}_{\text{app}} \leftarrow \mathcal{T}_{\text{app}} \cup \{\text{multi-session}\}$
end if
for all $t \in \mathcal{T}_{\text{app}}$ **do**
  if t is multi-session **then**
    $M \leftarrow$ gpt-4o-mini
  else
    $M \leftarrow$ gpt-4.1
  end if
  Use M to generate a question-answer pair (q_t, a_t) from $\mathcal{S}$, $\mathcal{D}$, and d_q when applicable
  Use M to verify whether a_t is supported by $\mathcal{S}$
  if verification accepts (q_t, a_t) **then**
    $\mathcal{Q}_{\text{syn}} \leftarrow \mathcal{Q}_{\text{syn}} \cup \{(q_t, a_t)\}$
  end if
end for
return $\mathcal{Q}_{\text{syn}}$

Importantly, the synthetic questions are generated solely from the oracle sessions, together with session dates and question dates when applicable. The original LongMemEval question and its reference answer are not provided during question generation or verification. The resulting questions are therefore used as preceding memory-use interactions rather than additional target questions. Each retained synthetic question is answered once and followed by one round of memory reconsolidation. The original LongMemEval question is then evaluated using the resulting memory state:

$$\mathcal{G}^{(0)} \rightarrow q_1 \rightarrow \mathcal{G}^{(1)} \rightarrow \cdots \rightarrow q_k \rightarrow \mathcal{G}^{(k)} \rightarrow q_{\text{target}},$$

where $k \leq 6$ and q_{target} denotes the original LongMemEval question. This construction provides preceding interactions for evaluating continual memory evolution while preserving the original benchmark question as the final evaluation target.

Table 3 presents the baseline results obtained directly on original LongMemEval questions without preceding synthetic interactions, providing a standardized basis to isolate the impact of memory reconsolidation and facilitate fair comparison.

Prompt B.1: LongMemEval Question Augmentation Prompt

System Prompt You are a question generation expert for conversational AI memory evaluation. Your task: given chat history between a user and an AI assistant, generate ONE new question-answer pair.
STRICT RULES:
1. The question must be answerable ONLY from the provided conversation(s) - no external knowledge.
2. The answer must be concise and directly supported by the text. Follow the word limit specified in the user prompt.
3. Do NOT invent facts, dates, names, or numbers that are not in the conversations.
4. Use the question style specified in the user prompt - do NOT force any particular framing. Some types ask directly ("What degree did I graduate with?"), others ask with context ("In our previous conversation about X, ...").

5. Output ONLY valid JSON in this exact format: “question”: “<question text>”, “answer”: “<answer text>”
6. If you cannot generate a valid grounded question, output: “question”: null, “answer”: null

Single Session User Prompt Below is a conversation between a user and an AI assistant.
Generate a question asking about ONE SPECIFIC DETAIL that the USER stated in this conversation.
RULES:
1. The question must be about something the USER said, and not the assistant’s advice or recommendations.
2. Ask about a SPECIFIC, CONCRETE detail: a name, number, date, place, price, color, or other verifiable fact.
3. The answer must be a short, specific phrase ($\leq$ 20 words) directly from the user’s words.
Good question styles (direct, no preamble):
- “What did I pay for [X]?” / “How much did I spend on [X]?”
- “Where did I [buy/attend/go for] [X]?”
- “When did I [do X]?”
- “What was the name of [X] I mentioned?”
- “How many [X] did I [have/buy/do]?”

Conversation: {session_text}
Pick one specific concrete USER-stated detail. Output JSON only.

Multi-Session Prompt Below are MULTIPLE conversations (sessions) between a user and an AI assistant, presented in chronological order.
Your task: generate ONE question whose answer REQUIRES reading AT LEAST TWO different sessions.
REQUIRED question patterns - pick the one that fits the content best (TYPE A/B are most common):

TYPE A - COUNT: something that accumulates across sessions
Q: “How many [X] have I [bought/visited/worked on] in total?”
A: a specific number (e.g. “3”, “five”)
Example oracle:“How many plants did I acquire last month?” $\rightarrow$ “3”
Example oracle: “How many model kits have I worked on or bought?” $\rightarrow$ “five”

TYPE B - SUM: numbers spread across sessions that add up
Q: “How much total [money/time] have I spent on [X]?”
Q: “How many [days/hours] did I spend on [X] in total?”
A: a specific total (e.g. “$185”, “15 hours”, “8 days”)
Example oracle: “How much total money have I spent on bike-related expenses since the start of the year?” $\rightarrow$ “$185”
Example oracle: “How many days did I spend on camping trips in the United States this year?” $\rightarrow$ “8 days”

TYPE C - CROSS-REFERENCE: a detail from session A only makes sense with context from session B
Q:“What [detail] on the [day/occasion] when I also [event from another session]?”
A: the specific detail
Example oracle: “What time did I go to bed the night before my doctor’s appointment?” $\rightarrow$ “2 AM”

TYPE D - MOST RECENT / FIRST: the latest or first occurrence of something across sessions
Q: “What was my most recent [X]?”
Q: “Where did I go on my most recent [trip/visit]?”
A: the name of the most recent/first item
Example oracle: “Where did I go on my most recent family trip?” $\rightarrow$ “Paris”

STRICT RULES:
1. Answer must be a SHORT SPECIFIC FACT: a number, dollar amount, time, or short phrase. NEVER a narrative.
2. Remove any one session $\rightarrow$ question must become unanswerable. If still answerable, pick different facts.
3. Do NOT ask about anything that exists entirely within one session.
4. Do NOT invent facts not in the sessions.
5. Answer $\leq$ 20 words.

FORBIDDEN (will be rejected):
$\times$ “What did I say/mention about X?” - answer becomes a narrative
$\times$ “What happened with X across our conversations?” - same problem

Sessions: {sessions_text}

```
Step 1: Scan sessions for facts that only make sense combined (counts across sessions, totals,
cross-session references).
Step 2: Check: “Is the answer a specific number, sum, or brief phrase?” If no → find different
facts.
Step 3: Check: “Readable from one session alone?” If yes → find different facts.
Step 4: Output JSON only: “question”: “...”, “answer”: “...”
```

B.5 Evaluation Details

Due to the limited availability of reproducible implementations and the substantial computational cost of reproducing all baselines, we reuse officially reported results whenever possible. To ensure a consistent comparison, we carefully align the backbone model, judge model, and evaluation procedure across methods. Specifically, all compared methods use ‘gpt-4o-mini” as the backbone model, as specified in their respective papers or implementations, and all evaluation judgments are performed by ‘gpt-4o-mini” based on the corresponding open-source evaluation implementations.

For LoCoMo, we use the evaluation prompt and scoring procedure provided in the open-source implementation of MemOS (Li et al., 2025), and follow the same judgment logic as MAGMA (Jiang et al., 2026a). For LongMemEval, we use the official evaluation prompts and scoring procedure provided by the benchmark. Together, these choices ensure a consistent evaluation setting across the compared methods.

In the main results, the performance of MIRIX and Mem0 is taken from the MemOS paper (Li et al., 2025), while the results of MAGMA and Nemori are adopted from the MAGMA paper (Jiang et al., 2026a). We reproduced the A-Mem results under the same experimental settings.

Prompt B.2: LLm Judge Prompt

```
You are an expert grader that determines if answers to questions match a gold standard answer.
Your task is to label an answer to a question as ’CORRECT’ or ’WRONG’. You will be given the
following data:
(1) a question (posed by one user to another user),
(2) a ’gold’ (ground truth) answer,
(3) a generated answer,
which you will score as CORRECT/WRONG.

The point of the question is to ask about something one user should know about the other user
based on their prior conversations. The gold answer will usually be a concise and short answer
that includes the referenced topic, for example: Question: Do you remember what I got the last
time I went to Hawaii? Gold answer: A shell necklace The generated answer might be much longer,
but you should be generous with your grading - as long as it touches on the same topic as the
gold answer, it should be counted as CORRECT.

For time-related questions, the gold answer will be a specific date, month, year, etc. The
generated answer might be much longer or use relative time references (like “last Tuesday” or
“next month”), but you should be generous with your grading - as long as it refers to the same
date or time period as the gold answer, it should be counted as CORRECT. Even if the format
differs (e.g., “May 7th” vs “7 May”), consider it CORRECT if it’s the same date.

Now it’s time for the real question:
Question: {question}
Gold answer: {gold_answer}
Generated answer: {response}

First, provide a short (one sentence) explanation of your reasoning, then finish with CORRECT or
WRONG. Do NOT include both CORRECT and WRONG in your response, or it will break the evaluation
script.

Just return the label CORRECT or WRONG in a json format with the key as “label”.
```

Algorithm 2 Cognitive Graph Memory Organization

Input: Observation o_t, Conversation Summary s_t, Recent Updates c_t, Memory Graph $\mathcal{G}$
Output: Updated Memory Graph $\mathcal{G}$

$X \leftarrow \textsc{ExtractMemoryUnits}(o_t, s_t, c_t)$
for x in X **do**
  $N \leftarrow \textsc{RetrieveSimilarAndRecent}(x)$
  $a \leftarrow \textsc{PredictOperation}(x, N)$
  $v \leftarrow \textsc{Execute}(a)$
  $E \leftarrow \textsc{InferRelations}(v, N)$
  $\textsc{UpdateGraph}(E)$
end for
return $\mathcal{G}$

Algorithm 3 Retrieval via Strategy Atom Combination

Input: Query q, Memory Graph $\mathcal{G}$
Output: Evidence Set $\mathcal{V}_q$, Activated Subgraph $\mathcal{G}_q$

$\pi_{seed} \leftarrow \textsc{ComposeSeedStrategy}(q)$
$R \leftarrow \emptyset$
for each retrieval plan $p \in \pi_{seed}$ **do**
  $R \leftarrow R \cup \textsc{SeedRetrieve}(\mathcal{G}, p)$
end for
while not $\textsc{StopRetrieval}(R)$ **do**
  $F \leftarrow \textsc{SelectFrontier}(R)$
  for each node $v \in F$ **do**
    $a_v \leftarrow \textsc{ComposeExpandStrategy}(v, R, q)$
    $N \leftarrow \textsc{Expand}(v, a_v)$
    $\textsc{UpdateAccessScore}(N)$
    $R \leftarrow R \cup N$
  end for
end while
$\textsc{Rerank}(R, q)$
$\mathcal{V}_q \leftarrow \textsc{TopK}(R)$
$\mathcal{G}_q \leftarrow \textsc{InduceSubgraph}(\mathcal{G}, \mathcal{V}_q)$
return $\mathcal{V}_q, \mathcal{G}_q$

C Additional Analysis

LongMemEval contains substantially different numbers of questions across categories. In particular, the single-session preference category contains only 30 questions, so one additional error changes the category accuracy by 3.33 percentage points. The reported REALM accuracy of 36.66% therefore corresponds to 11 correct answers out of 30, while the corresponding result without reconsolidation, 30.00%, corresponds to 9 correct answers. This granularity makes the category-level estimate sensitive to a small number of examples and should be considered when interpreting differences with other methods. However, the small sample size does not by itself explain the low absolute performance.

Unlike LoCoMo, each LongMemEval sample contains only one original target question. We introduce synthetic preceding interactions to create opportunities for memory evolution, but these interactions are constrained by the content and structure of the original sessions. In particular, not every sample can support the generation of all six interaction types, and most samples ultimately admit only two or three valid preceding interactions. Consequently, the augmented protocol provides fewer opportunities for repeated target-level interactions and memory evolution than LoCoMo, limiting the extent to which long-horizon topology adaptation can be observed within an individual sample. This is primarily a characteristic of the available evaluation data and augmentation setting rather than a limitation specific to REALM. Accordingly, we use LongMemEval mainly to examine

Algorithm 4 Feedback-Driven Topology Evolution

Input: Question q, Retrieved Subgraph $\mathcal{G}_q$, Feedback f, Memory Graph $\mathcal{G}$
Output: Updated Graph $\mathcal{G}$

$T \leftarrow \textsc{InferTopicStructure}(q, \mathcal{G}_q)$
$P^{key}, P^{noise} \leftarrow \textsc{IdentifyGroup}(q, \mathcal{G}_q, f, T)$
$\mathcal{D} \leftarrow \textsc{AgentEdit}(T, P, \mathcal{G}_q)$
for all $d = (v_i, v_j, a, r, c) \in \mathcal{D}$ **do**
  if $a =$ `create` **then**
    $\textsc{AddEdge}(v_i, r, v_j, \gamma c)$
  else
    if $a =$ `strengthen` **then**
      $w_{ij} \leftarrow w_{ij} + \gamma c(1 - w_{ij})$
    else if $a =$ `weaken` **then**
      $w_{ij} \leftarrow w_{ij} - \gamma c w_{ij}$
    end if
  end if
end for
return $\mathcal{G}$

whether the proposed memory mechanism generalizes beyond the LoCoMo setting, rather than as a fully controlled long-horizon evolution benchmark.

To better understand the remaining errors, we conducted a qualitative inspection of retrieval and answer-generation traces and distinguish three primary failure modes. A *retrieval miss* occurs when the supporting evidence is absent from the retrieved subgraph, including cases where the initial seed retrieval fails to identify the relevant topic. An *evidence-utilization error* occurs when supporting evidence is retrieved but is not correctly used in the final answer. A *harmful-update case* occurs when the system without reconsolidation answers correctly, whereas the system with reconsolidation answers incorrectly after a preceding topology update changes the relevant retrieval path. These categories separate failures of initial memory access from failures of reasoning and failures caused by subsequent memory evolution.

The single-session preference category is particularly affected by retrieval misses. Our trace inspection indicates that many incorrect cases fail at the initial topic-grounding stage: the seed retrieval does not identify the relevant preference-related topic, and subsequent graph expansion therefore cannot recover the required evidence. This reveals an upstream retrieval-coverage bottleneck that local reconsolidation cannot fully resolve when the relevant subgraph is never activated. Reconsolidation nevertheless improves this category by 6.66 points in the ablation study, suggesting that it can help when the relevant evidence is already reached, while the low absolute score shows that topology evolution does not replace reliable initial topic grounding.

The other categories exhibit different limitations. Single-session assistant and single-session user questions achieve relatively high accuracy and show limited gains from reconsolidation, as the relevant evidence is often localized within a small portion of the conversation. In such cases, the retrieved subgraph already provides sufficient structural coverage, leaving limited room for topology evolution to improve access to the answer. Multi-session and temporal questions are more dependent on jointly retrieving and connecting evidence across sessions, entities, or time. When only part of the required evidence is retrieved, the activated subgraph may remain incomplete, and reconsolidation cannot recover evidence that was never activated. Even when the relevant evidence is retrieved, missing cross-session or temporal connections can still limit the benefit of local topology updates. Knowledge-update questions benefit more from reconsolidation, as their answers often depend on distinguishing successive or superseded information. Topology evolution can strengthen relations between related pieces of evidence and improve access to the more recent information, making these questions more directly aligned with the benefits of reconsolidation.

Overall, reconsolidation is most useful when the relevant evidence has already entered the activated subgraph and the remaining challenge is to organize or connect it, so that it can be discovered collectively during retrieval. Its benefit is inherently limited when the bottleneck lies in initial retrieval coverage, downstream reasoning abilities, or when the required evidence is already sufficiently lo-

calized. More broadly, the limited number of valid preceding interactions in the LongMemEval augmentation constrains opportunities for such topology evolution, which should be taken into account when assessing the relatively modest gains.

D ALGORITHMS FOR REALM

To facilitate understanding and reproducibility, we provide the complete pseudo-code for the proposed REALM framework. The algorithms correspond to the three stages introduced in the method section: memory organization, memory retrieval, and memory reconsolidation. They summarize the control flow and interactions among the LLM, memory graph, and retrieval modules while omitting implementation-specific details for readability. The pseudo-code is presented in Algorithms 2, 3, and 4, respectively.

E RETRIEVAL DETAILS

E.1 STRATEGY ATOMS

This section walks through the three retrieval prompts (Prompt F.4, F.5, F.6) by option, describing how each option that actively affects retrieval is executed.

Seed Retrieval Planner (Prompt F.4). Each seed retrieval plan $p_i = (Q_i, K_i, T_i, \tau_i)$ involves three key decisions: matching mode that determines how Q_i and K_i are used to score candidate nodes, T_i and τ_i are used to filter node types and time, respectively. Multiple seed plans issued for one question are executed independently and merged into a single seed set by node id.

- **matching_mode** is determines how Q_i and K_i are used to score candidate nodes:
  - `query_match`. Only Q_i is embedded and matched against node embeddings by cosine similarity; keywords are not used.
  - `keywords_match`. Only K_i is matched against nodes through lexical, inverted-index keyword search; the query embedding is not used.
  - `hybrid_match`. Both signals are computed for every node and combined as a weighted sum (embedding weight 0.6, keyword weight 0.4 by default), so a node found by either signal can be recalled.
- **allowed_node_types** (T_i) is applied as a memory node type filter in all three modes, but is not strictly enforced. If restricting to T_i together with the score threshold would leave no candidates, the filter is relaxed in two steps: first the score threshold is dropped while keeping T_i, and then if the result is still empty, T_i itself is dropped. Therefore, seed retrieval never returns an empty set.
- **time_range** (τ_i) is only enforced when matching mode is `hybrid_match`: a node is kept only if its own time interval overlaps τ_i, nodes outside this window are excluded outright, and a node carrying no resolvable timestamp is treated as always inside the window and kept by default.

Retrieval Sufficiency Controller (Prompt F.5). Before each expansion round, this atom inspects the current retrieved nodes set and returns two fields that control the loop:

- **is_enough**. If true, retrieval terminates immediately with the current retrieved memory nodes.
- **nodes_to_expand**. If `is_enough` is false, this list of node ids becomes the next expansion frontier F_t. If the list is empty or none of its ids belong to the current nodes set, the frontier instead defaults to the highest-scoring nodes that have not yet been expanded.

Graph Expansion Policy (Prompt F.6). For each frontier node $v \in F_t$, this atom produces an expansion action $a_v = (\text{mode}, \text{predicate}, \text{decay}, \text{inhibit})$ together with `score_threshold` and `reasoning`. Every candidate edge e reachable from v, with endpoint node u, is scored as

$$score(e, u) = w_{sem} \cdot \text{sim}(e, r) + w_{edge} \cdot \omega_e + \delta_{temporal}, \tag{15}$$

clipped to $[0, 1]$, where $\text{sim}(e, r)$ is the cosine similarity between the embedding of edge e's description and the embedding of the **reasoning** text r, ω_e is the current edge weight, $\delta_{temporal}$ is the

penalty contributed by **decay** (zero when disabled), and $w_{sem} = 0.6, w_{edge} = 0.4$ by default. A candidate is discarded before scoring if it revisits a node already on the current path, has already been reached from another seed, or is excluded by **predicate**; after scoring, it is further discarded if its score falls below `score_threshold`.

- **predicate** (`layer_predicate_include`): restricts candidate edges to a given relation layer sets from {logical, causal, taxonomic, associative}, optionally one predicate within that layer, and a traversal direction (forward, backward, or both); this is a hard filter applied before scoring.
- **decay** (`temporal_decay`): is used to compare a candidate node's timestamp against a reference window (time $\pm$ Δt). `hard` mode drops out-of-window candidates entirely; `soft` mode instead applies a continuous penalty that grows with the distance beyond the window boundary. Nodes lacking timestamp information are kept with a fixed penalty under `missing_time_mode=keep`, or dropped outright under `missing_time_mode=drop`.
- **inhibit** (`conflict_inhibit`): is used to discard any candidate edge whose predicate is `contradicts` before scoring when enabled.

The **mode** atom determines how candidates in the expansion stage are selected and carried into the next hop:

- `subgraph_beam`. At each hop, aggregate all candidate nodes from the active frontiers, remove duplicates based on the target node ID (retaining the entry with the highest score), and admit them in order of score until a single node budget shared by the whole seed (`subgraph_max_nodes`) is exhausted. The frontier is then completely replaced by the newly accepted nodes.
- `path_search`. Each active path is extended independently. The generated paths are deduplicated based on their terminal nodes (retaining the path with the highest cumulative path score, calculated as the average hop score minus a fixed penalty of 0.03 for each additional hop), and only the top-`path_top_k` paths remain active for the next hop, for a maximum of `path_max_depth` hops. Paths that are pruned at a particular hop will not be reconsidered.

F PROMPTS

For completeness, we provide the detailed prompt templates across all three functional stages of our framework:

- Memory Organization:
  - Memory Unit Construction (Prompt F.1): Extracts memory units and updates dialogue summaries.
  - Memory Operation Selection (Prompt F.2): Decides node insertion, merging, or skipping.
  - Memory Relation Prediction (Prompt F.3): Determines edge operations and semantic relations between nodes.
- Retrieval Stage:
  - Seed Planning (Prompt F.4): Selects retrieval modes and constraints to identify anchor nodes.
  - Sufficiency Assessment (Prompt F.5): Evaluates evidence completeness and identifies candidate expansion nodes.
  - Retrieval Expansion (Prompt F.6): Defines traversal modes and edge constraints for graph propagation.
- Topology Evolution:
  - Topology Refinement (Prompt F.7): Guides local graph topology evolution based on retrieved subgraphs.

Prompt F.1: Memory Construction

You are a memory construction agent of an agent memory system. You must perform two tasks:

Task A: Memory Unit Generation. Based on your relevant **past memories**, create memory units for the **Latest Information**.
Task B: Dialogue Summary Generation. Based on the latest information, update the **current conversation summary**.

The past memories include a summary of the past conversation and related conversation content. The latest information is the new input information. When generating new memory units, you need to assess their relationship to past memories. For example, the new information might answer a question raised before, or it might serve as a supplement or extension to a past memory, and so on. You need to establish this relationship within the memory units you've extracted. Use the principle of the Fewest Memory Nodes: represent the **Latest Information** clearly and completely with the fewest nodes.

Task A: Memory Unit Generation

Step 1. Extract Information. Referring to the supplementary information provided by past memories, identify the key points in the latest information and represent them using the most appropriate of the following four levels of granularity:
- entity: a referable object such as a person, place, object, or concept.
- event: a single action, interaction or occurrence, describing something that happened at a specific time
- episode: a multi-step experience with context and outcome, lasting for a period of time
- fact: a stable and general statement about somebody and something, or a universal rule and knowledge, derived or deduced from latest information

INSTRUCTIONS:
- The past memories have already been saved and does not need to be saved again.
- Avoid mixing multiple semantics in one unit.
- It is allowed to generate multiple units if needed
- Ensure that each unit is semantically coherent and complete

Step 2. Construct Memory Units. Follow the structure below to construct memory units:
entity: name; aliases (a list of alternative names, NOT pronouns); category (person | object | place | concept | etc); role (describe the role or significance of this entity).
event: name (general event name); participants (a list of relevant entity names); description (a description of the event); time (exact happened time of described event: none | xxxx-xx-xx).
episode: name (general episode name); description (a description of the episode, contains the why it happened and the outcome); keywords (a list of keywords for the episode, not more than 3); time_range (exact time range of described episode: none | xxxx-xx-xx to xxxx-xx-xx).
fact: statement (a clear and universal factual statement); keywords (a list of keywords for the fact, not more than 3).

INSTRUCTIONS:
- When declaring each field in a memory unit, you must specify the fully qualified name of the element.
- Each of these memory units should be able to be semantically understood independently.

Task B: Dialogue Summary Generation
Based on the latest information, update the **current conversation summary**. This summary serves as your long-term working memory as a memory construction agent, helping you understand the context and progress of the information currently being processed. The **current conversation summary** should include content such as how the conversation topic has evolved and any ongoing, unfinished tasks, to provide a high-level overview of the conversation's progress.

Keep the content concise and general; avoid using any demonstrative pronouns, and do not exceed 200 words.

GOOD example slices:
- "A is asking B about winter travel plans."
- "They are discussing John's son's academic situation."

Prompt F.2: Node Operation Decision

You are maintaining a memory graph. Your task is to decide how to handle a new memory unit.

Task Overview Given a new memory unit and candidate nodes (retrieved by similarity and/or recency), you need to:

1. Decide the operation type.
- add: Create a new node for this memory unit
- modify: Merge this memory unit into an existing candidate node
- skip: Skip storing because an existing node already completely covers this content

2. Output the final node information. (for both "add" and "modify"):
- You may adjust/refine the node's metadata based on context
- For "modify": output the merged node information
- For "add": output the new node information (can be same as input or refined)

3. Identify related candidates.
- Select candidates that have semantic and logical relationships with the final node
- These will be used for edge creation in the next step
- You do NOT need to specify edge types here, just identify which candidates are related

Decision Criteria
Choose **modify** only if all of the following conditions are met:
- The new information refers to the SAME underlying entity/event/fact
- It only adds detail, clarification, or minor correction
- It does NOT introduce: a new state, a temporal change, or a contradiction
- Merging will NOT cause loss of previously stored information and the memory unit information (their content are duplicated)

Choose **skip** if:
- An existing node already completely covers the content
- Storing this would create pure duplication

Otherwise, choose **add**.

Important Principle
When uncertain, prefer "add" over "modify", since an incorrect merge may cause irreversible information loss.

Candidate Sources
Candidates come from two sources (marked in the input):
- **similarity**: Retrieved by semantic similarity to the new memory unit
- **time**: Recently modified/created nodes (temporal proximity)
- **similarity+time**: Retrieved by both methods

Consider both similarity scores and temporal context when making decisions.

Prompt F.3: Edge Connection Decision

You are maintaining a memory graph. Your task is to decide how to establish edges between a target node and related nodes.

Task Overview: Given a target node and its related candidate nodes (some may already have edges to the target), decide:
- For each candidate: what edge operation to perform
- Edge operations: "add" (new edge), "modify" (change existing edge), "delete" (remove edge), "no_change" (keep as is)

Edge Predicates (Organized by Type)
1. Logical
- **contradicts(A,B)**: A and B are logically mutually exclusive. When A is detected, the activation of B should be suppressed.
- **implies(A,B)**: If A holds, then B must hold.
- **constrains(A,B)**: The existence of A restricts the boundaries or execution parameters of B.
- **verifies(A,B)**: A serves as evidence for B, increasing confidence in B.

2. Causal
- **precedes(A,B)**: Pure temporal sequence; A occurs before B, with no direct causality.
- **enables(A,B)**: A is a necessary condition for B to occur (necessary Condition). Without A, B cannot be initiated.
- **triggers(A,B)**: The occurrence of A directly causes B (Sufficient Condition).
- **results_in(A,B)**: B is the state change resulting from the execution of A (State Transition).

3. Taxonomic
- **comprises(A,B)**: A whole-part relationship. A is composed of multiple parts, including B.
- **instantiates(A,B)**: A relationship between abstract class A and concrete instance B.

- **summarizes(A,B)**: A is a high-level semantic compression of B (typically episodes or events).

4. Associative
- **correlates(A,B)**: Semantically highly similar or statistically strongly correlated.
- **supersedes(A,B)**: A is the latest version or correction of B; B should be considered obsolete or historical.
- **analogous_to(A,B)**: An analogical relationship. A and B are similar in structure or function, although they belong to different domains.
- **contextualizes(A,B)**: A provides necessary background information or contextual clarification for B.

Decision Guidelines
For add: Create a new edge only when the relationship is clear. Specify the predicate, weight, and description.
For modify: Update an existing edge when the predicate is more appropriate or the weight should be adjusted. Output the complete updated edge.
For delete: Remove an edge only when the relationship is clearly invalid. Prefer **no_change** when uncertain.
For no_change: Keep the existing edge unchanged when it remains valid.

Direction Rules
- For single-direction predicates: source -> target indicates the relationship direction
- For bidirectional predicates (contradicts, correlates, analogous_to): the system will automatically create reverse edges

Weight Guidelines: Weight represents the certainty/strength of the relationship [0, 1]:
- 0.9-1.0: Very confident, explicit relationship
- 0.7-0.8: Confident, clear relationship
- 0.5-0.6: Moderate confidence, implicit relationship
- 0.3-0.4: Low confidence, weak relationship
- 0.0-0.2: Very uncertain

Prompt F.4: Seed Retrieval Planner

Task Analysis the question, thinking about the key entities, contents, potential logical chains, potential related factors, etc., and generate a strategy to locate the anchor points on the memory graph.

Choose one mode from following three modes:
- query_match: generate a new query based on the question, and only uses embedding semantic similarity to search the memory graph. It is good at recalling memories expressed with different wording, implicit context, paraphrased situations, and broad semantic targets.
- keywords_match: generate a list of keywords based on the question, and only uses lexical keyword matching to search the memory graph. It is good at preserving precise surface anchors such as names, places, organizations, project, object, event, and quoted phrases.
- hybrid_match: generate a new query and a list of keywords based on the question, and take both scores into consideration to search the memory graph. It is good when initial recall should preserve exact anchors while also matching the semantic condition.

Return strict JSON keys:
- allowed_node_types: the types of the nodes to retrieve.
-time_range: the temporal constraint for retrieval.

Allowed Node Types Candidates:
- entity: A referable object such as a person, place, object, or concept.
- event: a single action or something happened at a specific time
- episode: a multi-step experience with context and outcome, lasting for a period of time
- fact: stable, complete, generalizable and explicit statement derived or deduced from NEW CONTENT

Prompt F.5: Retrieval Sufficiency Controller

You are a retrieval controller in a graph memory system. Decide whether the displayed candidate nodes already contain enough information to answer the question.

First decide global sufficiency using all displayed candidate nodes together. Return “is_enough”: “true” when at least one candidate node, or the combination of multiple candidate nodes, contains

the answer and you can directly answer the question from the shown evidence. It is NOT required that every candidate node can answer the question.

When “is_enough” is “true”, “nodes_to_expand” must be [] and “nodes_to_skip” must be []. Do not expand other nodes just because they do not contain the answer.

Return “is_enough”: “false” only when all displayed candidate nodes together still do not contain the answer. In that case, choose which candidate nodes should be used as starting points for the next graph expansion round.
In Graph Memory System, content-related nodes are connected by various semantic and logical relationships.
- nodes_to_expand: Select a node only if it is relevant to the topic of the question, contains partial evidence, provides a reasoning bridge to the answer, is located in a graph neighborhood that may contain the answer, or holds the potential to lead to evidence. Do not expand a node

just to confirm an answer that is already explicitly stated in that node. Only select the nodes with the highest expansion potential, no more than 5 nodes.

Prompt F.6: Graph Expansion Policy

Given the current retrieval state, generate graph expansion strategy for each selected seed node. The available edge predicates are identical to those defined in the **Edge Connection Decision** prompt.

Strategy of Each action:
Necessary Parameters:
- seed_id: the id of the seed node.
- expansion_mode: choose one of [“subgraph_beam”, “path_search”].
  - subgraph_beam: builds a compact relevant subgraph by globally selecting high-quality candidates. Use this when the question needs broad evidence aggregation, related facts, constraints, or contextual support.
  - path_search: searches coherent reasoning paths. Discarded paths/candidates must not be reconsidered later. Use this when the question needs chain-like evidence, causal/logical progression, or one consistent explanation route.
- layer_predicate_include: the layer-predicate-direction constraints to include. Each item is a dictionary with the format of {“layer”: str, “predicate”: str, “direction”: “forward|backward|both|none”}.
  - layer must be one of [“logical”, “causal”, “taxonomic”, “associative”].
  - predicate must belong to the selected layer. If you only want to restrict by layer, use an empty predicate.
- score_threshold: minimum candidate score to keep. Candidates below this threshold should be filtered before adding them to a subgraph or path.

Mode-specific Parameters:
- For expansion_mode = “subgraph_beam”:
  - subgraph_max_nodes: maximum number of unique non-seed nodes to collect for this seed under this action. Use a small number for focused retrieval and a larger number only when the question needs broad supporting evidence.
- For expansion_mode = “path_search”:
  - path_max_depth: maximum path length measured by number of edges from the seed. This is the hard upper bound for coherent path exploration.
  - path_top_k: maximum number of active paths to keep at each hop. Use 1 for a single greedy path, and a larger value only when multiple plausible reasoning chains should be explored.

Optional Parameters:
- temporal_decay: optional time constraint operator. Use this only when the question requires explicit temporal constraints.
  - strict JSON format:

```
{
  "time": "YYYY|YYYY-MM|YYYY-MM-DD|YYYY-MM-DD HH:MM|[start,end]",
  "delta_t": "supports value+unit, e.g. {"value": 7, "unit": "days"}",
  "mode": "hard | soft",
  "missing_time_mode": "keep | drop"
}
```

  - semantics:
    - time is determined by the question.

```
        - delta_t is the tolerance window (time ± delta_t): in-window => no penalty.
        - hard: out-of-window nodes should be treated as filtered.
        - soft: out-of-window nodes are not filtered, but decayed by temporal penalty.
        - missing_time_mode = keep: keep node with a fixed penalty.
        - missing_time_mode = drop: drop node directly.
- conflict_inhibit: the flag of the conflict inhibit.
- reasoning: describe what content this strategy is intended to find.
```

Prompt F.7: Topology Update

You are a topology auditor for a graph memory system. You are tasked with analyzing a retrieved memory subgraph and identifying how its topology should be refined.

Task Objectives
This is a local topology reconstruction task. You will be provided with a set of recalled nodes and the existing edges between them. Your task is to determine whether new edges should be established between these nodes or whether the weights of existing edges should be adjusted. The purpose of these operations is to make the relationships between nodes more logically rigorous and complete, and to facilitate future retrieval.

The provided questions and answers serve as references to assist you in reconstructing the local topology. You can analyze the contribution of these nodes to the questions to better understand the relationships between them. Your ultimate goal is not to enhance the connections within this specific problem, but to improve future graph access: key evidence should be accessible through a reliable logical structure, misleading connections should have their weights reduced, and connections between nodes unrelated to the problem should be established only if their relationships are clearly defined.

Context
Allowed Edge Schema: (Type-Predicate)

{edge_type_schema}

Confidence Guidelines
For every operation, output confidence, not manual weight delta. Confidence represents how certain you are that this operation is useful and correct for future graph retrieval [0, 1]:

{confidence_rule}

For add_edge, confidence will be used directly as the new edge weight. For strengthen_edge/weaken_edge, the system will convert confidence to the weight change; do not output delta.

Required Analysis Workflow
Before deciding edits, explicitly analyze the recalled nodes in the JSON output:
1. Determine the core topic represented by the recalled nodes. The input question should only be treated as a reference signal for identifying the topic. If the answer to the question is incorrect, focus instead on the underlying topic that the recalled nodes collectively imply.
2. Identify which nodes constitute the key evidence required to answer the question correctly, or are directly relevant to the identified topic. Input them at “gold_supporting_node_ids”.
3. Identify which nodes provide supporting context, including but not limited to: background information, constraints, temporal information, entity disambiguation, or other information necessary for understanding the topic. Input them at “supporting_associated_node_ids”.
4. Identify which nodes are irrelevant to the answer or topic, or may introduce misleading directions during retrieval and reasoning. Input them at “misleading_node_ids” or “unrelated_node_ids”.
5. Based on the above analysis, propose topology refinement actions. In particular:
  - strengthen predicate relations among generally relevant nodes,
  - weaken the influence of topic-irrelevant or misleading nodes,
  - add missing connections between nodes that should be related through valid predicates but are currently disconnected,
  - improve the logical coherence of the local subgraph to better support future retrieval and reasoning.

Constrains 1. Only create new edges between recalled nodes, and only adjust the weights of existing edges between these nodes.
2. Preserve the existing graph structure. Do not delete edges or modify node content.

```
3. Use only predicates from the provided allowed edge predicate set.
4. When adjusting an existing relationship, use strengthen_edge or weaken_edge and reference a
valid edge_id from the existing graph.
5. Add a new edge only when the relationship is strongly supported by the available evidence
from the recalled nodes. If the evidence is ambiguous, do not add the edge.
6. Adjust edge weights only when there is clear evidence that the relationship strength should
change and that doing so is likely to improve future memory retrieval.
```

F.1 AI USAGE AND ETHICS STATEMENT

Generative AI tools were used for finding related works, editing language and checking grammar during paper writing. All scientific content, including the research ideas, methodology, experiments, analyses, and conclusions, was developed and verified by the authors, who take full responsibility for the paper.