LGMEM: LIGHTWEIGHT GRAPH MEMORY FOR LONG-TERM CONVERSATIONAL AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Agents engaged in sustained interactions need to accurately recover dispersed and evolving facts from growing dialogue histories. Existing structured memory systems face two challenges: (1) how to construct memory cost-effectively while preserving fine-grained evidence; and (2) how to reliably obtain sufficient evidence from memory. To address these challenges, we introduce **L**ightweight **G**raph **Mem**ory (LGMem), a memory framework that combines efficient memory construction with reflection-guided iterative retrieval. During construction, LGMem uses lightweight entity and keyword extraction to incrementally connect original sentences through shared anchors, building graph memory that retains raw dialogue evidence without calls to large language models (LLMs). At query time, LGMem uses local graph propagation to retrieve candidate evidence across sessions and stateful reflection to identify evidence gaps and iteratively refine queries to gather missing evidence. Experiments across multiple benchmarks show consistent QA gains over strong baselines. On LongMemEval-S and BEAM-100K, LGMem improves QA scores over the strongest respective baselines by 8.0% and 9.8%, while reducing total token consumption for memory construction and access by 75.8% and 84.5%, demonstrating a better trade-off between QA performance and token cost.

1 INTRODUCTION

As interactions continue over time, agents need to remember user facts and preferences across sessions, track information updates, and combine dispersed clues to answer current questions (Zhang et al., 2025; Wu et al., 2025). Long-term memory must support not only the storage of interaction histories but also continuous updates and question-driven access (Du et al., 2025). As dialogue histories grow, evidence becomes dispersed across sessions and speakers, while information updates and conflicting records complicate cross-session question answering (Maharana et al., 2024; Wu et al., 2025; Tavakoli et al., 2026). Feeding the entire history into a long-context model increases processing costs as the history grows, while a larger context window does not ensure reliable use of distant evidence (Liu et al., 2024). Consequently, long-term dialogue QA requires efficient retrieval and integration of question-relevant evidence from accumulated histories.

Existing agent memory methods manage growing interaction histories through compression, consolidation, and structured indexing. Summaries and compact states reduce the amount of information processed during subsequent access (Liu et al., 2026; Zhou et al., 2026). LightMem (Fang et al., 2026), for example, combines lightweight filtering with topic-grouped summarization and schedules long-term memory consolidation offline to reduce online processing costs. Multi-granularity indexes and graphs connect dispersed information to support retrieval across segments (Jiang et al., 2026; Zhuang et al., 2026). MemGAS (Xu et al., 2026b) associates memories at multiple granularities and adaptively selects the retrieval granularity for each query. During memory access, MRAgent (Ji et al., 2026) and REMem (Shu et al., 2026) collect answer evidence through iterative graph exploration and retrieval.

However, existing structured memory systems still face two challenges in memory construction and access. First, fact extraction, summarization, and memory consolidation based on large language models (LLMs) repeatedly process incoming information and revisit existing memory, accumulating model calls, token consumption, and processing time (Fang et al., 2026; Chhikara et al., 2025).

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.

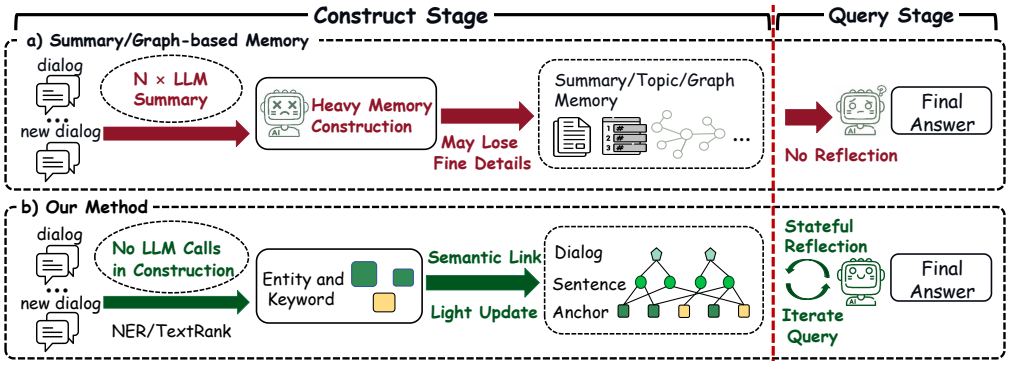


Figure 1: Comparison of existing memory methods and LGMem.

Because future queries are unknown during construction, compressing raw dialogue risks discarding details needed for later answers. Even a summary that accurately captures the main content can omit specific dates or conditions required by a query. Second, initial retrieval does not guarantee sufficient evidence for cross-session questions, especially when relevant facts and temporal relations are distributed across multiple sessions. To answer such questions, a memory system must assess the accumulated evidence, identify missing information, and guide subsequent retrieval accordingly.

We therefore introduce **L**ightweight **G**raph **Mem**ory (LGMem), a new memory framework that combines efficient, evidence-preserving construction with reflection-guided iterative retrieval. As shown in Figure 1, LGMem retains and organizes raw evidence during history ingestion and defers question-specific evidence selection and integration to query time. During construction, LGMem connects original sentences through shared entity and keyword anchors. It incrementally builds graph memory with sentence–segment membership links, preserving raw dialogue evidence, provenance, and temporal information without LLM calls. At query time, LGMem first retrieves candidate evidence across sessions through anchor activation and local graph propagation. It then uses stateful reflection to assess the evidence, track known information and remaining gaps in an explicit query state, and refine queries to iteratively gather missing evidence.

On LongMemEval-S and BEAM-100K, LGMem improves QA scores over LightMem and MemGAS, the strongest respective baselines, by 8.04% and 9.84%, while reducing total generative token consumption for memory construction and access by 75.80% and 84.49%. On BEAM-500K and BEAM-1M, LGMem improves QA scores over the strongest baseline, SimpleMem, by 15.14% and 17.43%, with corresponding token reductions of 66.57% and 82.16%. **The code is available at** `https://anonymous.4open.science/r/lgmem-anonymous-16B1`. Our main contributions are as follows:

- We introduce LGMem, a lightweight graph memory framework that incrementally organizes raw dialogue evidence into a unified anchor–sentence–segment graph. By using entity and keyword anchors, LGMem preserves fine-grained evidence and provenance without relying on LLMs during memory construction.
- We further introduce stateful reflection into LGMem's iterative memory access. By maintaining an explicit query state, LGMem assesses accumulated evidence, identifies unresolved gaps, and guides follow-up retrieval through lightweight local graph propagation.
- Extensive experiments on LongMemEval-S with two generator backbones and on BEAM with dialogue histories up to 1M tokens show that LGMem consistently improves QA performance while substantially reducing generative token consumption, demonstrating a favorable quality–efficiency trade-off.

2 Method

2.1 Overview

Problem setting. We study long-term dialogue QA over sequentially arriving sessions $\mathcal{D}_t$, each with a timestamp τ_t and speaker information. Given a question q and an optional reference time

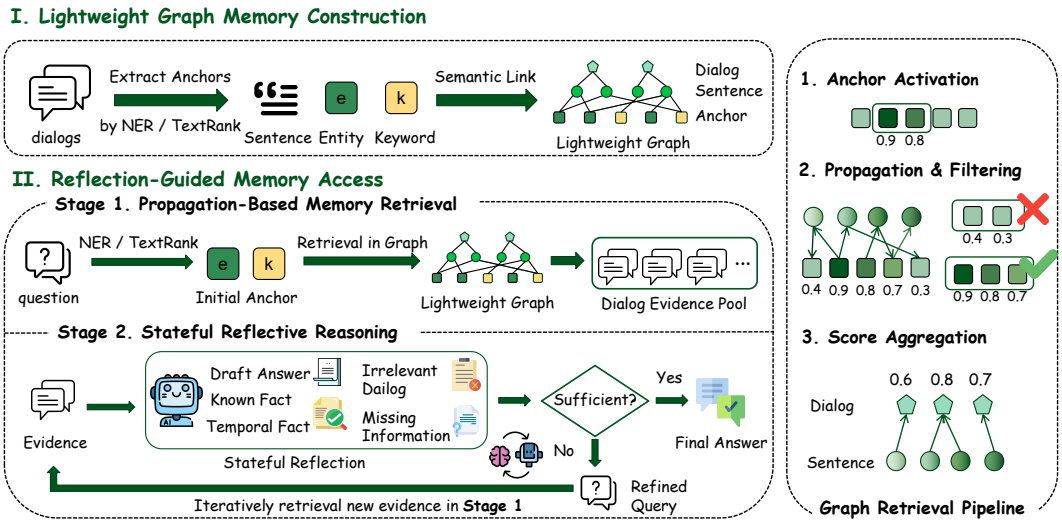


Figure 2: **Overview of LGMem. I. Lightweight Graph Memory Construction.** LGMem uses entity and keyword anchors to organize original sentences and their parent segments into an incrementally updated relation-free graph without LLM calls. **II. Reflection-Guided Memory Access.** LGMem retrieves dialogue evidence through anchor activation and local graph propagation, then uses stateful reflection to assess evidence sufficiency and refine queries to gather missing information before final answer generation.

τ_q, the system produces an answer $\hat{y}$ grounded in the dialogue history. Answering may require integrating facts across sessions and reasoning about temporal relations and information updates.

Pipeline overview. LGMem separates persistent memory construction from query-local iterative access, as illustrated in Figure 2. Each arriving session incrementally updates graph memory $\mathcal{G}_t$, while each question maintains an evidence pool and an explicit query state that guide retrieval until sufficient evidence supports answer generation. Sections 2.2 and 2.3 describe the two stages.

2.2 Lightweight Graph Memory Construction

Inspired by LinearRAG (Zhuang et al., 2026), LGMem represents dialogue history with three node types: segments d_k that preserve contiguous dialogue context, sentences s_j that provide fine-grained evidence, and anchors a_i that represent extracted entities or keyword phrases. These nodes form a unified, relation-free anchor–sentence–segment graph: anchors link to their source sentences, and each sentence links to its unique parent segment.

Evidence units and shared anchors. Each arriving session $\mathcal{D}_t$ is segmented into consecutive windows of dialogue turns, and each turn is split into sentences. Segment and sentence nodes retain raw text, session and speaker provenance, and the session timestamp τ_t. LGMem extracts anchors through two complementary lightweight methods: named entity recognition (NER) using spaCy (Honnibal et al., 2020) and sentence-level keyphrase extraction using TextRank (Mihalcea & Tarau, 2004). Entity anchors provide object-level links, while keyword anchors capture recurring activities, topics, and concepts. Shared anchors connect dispersed sentences, while sentence–segment membership provides access to the original context.

Incremental ingestion. Upon receiving a new session, LGMem appends its segment and sentence nodes and the corresponding links to the existing graph. Extracted entities and keyphrases are matched to existing anchor nodes of the same type, and a new anchor node is created only when no match is found. This incremental process updates the memory to $\mathcal{G}_t$ without LLM calls while retaining raw evidence for later queries.

2.3 REFLECTION-GUIDED MEMORY ACCESS

At query time, LGMem alternates propagation-based retrieval with stateful reflection: retrieval gathers candidate evidence, and reflection assesses the accumulated evidence and formulates follow-up queries for unresolved gaps.

2.3.1 PROPAGATION-BASED MEMORY RETRIEVAL

Given a retrieval query, LGMem activates relevant anchors and propagates their scores through neighboring sentences and anchors before aggregating sentence scores to rank their parent segments. Following shared anchors across hops enables retrieval to reach related evidence distributed across sessions beyond the initially activated concepts.

Query activation. Given a retrieval query q', which may be the original question or a follow-up query, LGMem extracts entity and keyword anchors. Let $\mathrm{sim}(x, y)$ denote the cosine similarity between text embeddings (Reimers & Gurevych, 2019), and let $\mathcal{U}_i(q')$ denote the query anchors of the same type as graph anchor a_i. Its relevance score is

$$r_i = \max_{u \in \mathcal{U}_i(q')} \mathrm{sim}(u, a_i). \tag{1}$$

LGMem selects up to K_a anchors with the highest relevance scores as seeds and assigns them initial activations $z_i^{(0)} = r_i$; all remaining anchors have zero initial activation.

Local propagation. At hop ℓ, each anchor whose activation $z_i^{(\ell-1)}$ reaches the propagation threshold θ_p selects up to K_p unvisited neighboring sentences with the highest similarities to q'. The threshold limits further traversal to sufficiently strong signals. Let $\mathcal{A}_j^{(\ell)}$ denote the expanding anchors that select sentence s_j at hop ℓ. Its score combines its query relevance with the strongest activation among these anchors:

$$h_j^{(\ell)} = \mathrm{sim}(q', s_j) \max_{a_i \in \mathcal{A}_j^{(\ell)}} z_i^{(\ell-1)}. \tag{2}$$

Each selected sentence passes its score to its neighboring anchors. Let $\mathcal{C}_i^{(\ell)}$ denote the selected sentences connected to anchor a_i. Its activation is updated by retaining the strongest score observed so far:

$$z_i^{(\ell)} = \max\left\{ z_i^{(\ell-1)}, \max_{s_j \in \mathcal{C}_i^{(\ell)}} h_j^{(\ell)} \right\}. \tag{3}$$

Anchors whose updated activation reaches θ_p expand at the next hop, allowing retrieval to follow entities or topics discovered in the selected sentences. Propagation stops when no anchors remain for further expansion or when the hop limit L is reached. The final score of sentence s_j is $h_j = \max_{1 \le \ell \le L} h_j^{(\ell)}$.

Segment ranking and evidence output. For each segment d_k, the graph evidence score $g(d_k)$ aggregates at most its top K_g supporting-sentence scores with a logarithmic length penalty, limiting the influence of long segments containing many weak matches. After separately normalizing graph evidence scores and dense query–segment similarities, LGMem computes the final segment score:

$$\mathrm{score}(d_k) = \widetilde{g}(d_k) + \lambda_d \widetilde{\mathrm{sim}}(q', d_k) + \lambda_r \rho(d_k). \tag{4}$$

The graph term captures associative evidence reached through shared anchors, while dense similarity preserves direct relevance to q'. The recency term $\rho(d_k)$ gives a modest preference to later evidence when subsequent conversations update earlier information.

The coefficients λ_d and λ_r control the contributions of dense similarity and dialogue recency. We set $\lambda_d = 0.15$ and $\lambda_r = 0.05$ in all experiments. LGMem returns the top-ranked segments together with high-scoring supporting sentences: segments preserve the surrounding dialogue context, while supporting sentences highlight the evidence identified through graph propagation. Each item retains a stable identifier and its session, speaker, and temporal provenance for subsequent reflection. Further configuration details are provided in Appendix B.1.

2.3.2 STATEFUL REFLECTIVE REASONING

Evidence pool and query state. Across retrieval rounds, the evidence pool $\mathcal{P}^{(r)}$ stores accumulated raw evidence, while the blocked set $\mathcal{B}^{(r)}$ records evidence excluded from subsequent retrieval. Evidence items are merged by stable identifier and ranked. Let $\mathcal{X}^{(r)}$ denote the resulting evidence view presented to the reflection module at round r.

LGMem organizes its current findings in an explicit query state $\mathcal{S}^{(r)} = (\widetilde{y}^{(r)}, \mathcal{F}^{(r)}, \mathcal{T}^{(r)}, \mathcal{M}^{(r)})$, consisting of a draft answer, evidence-supported facts, structured temporal records, and unresolved information gaps. Separating the persistent evidence pool from the explicit query state allows LGMem to retain raw evidence while tracking what has been established and what remains unresolved. Both persist across retrieval rounds, and state entries retain references to their supporting evidence.

Reflection and targeted retrieval. At round r, an LLM-based reflection procedure Φ takes the original question and its time reference, the current evidence view $\mathcal{X}^{(r)}$, the previous query state $\mathcal{S}^{(r-1)}$, and the query history $\mathcal{H}^{(r)}$, which contains all queries executed through that round. It returns an updated state, follow-up queries $\mathcal{Q}^{(r+1)}$, newly blocked evidence identifiers $\Delta\mathcal{B}^{(r)}$, an evidence-sufficiency judgment $b^{(r)} \in \{0, 1\}$, and a confidence score $c^{(r)} \in [0, 1]$:

$$\left(\mathcal{S}^{(r)}, \mathcal{Q}^{(r+1)}, \Delta\mathcal{B}^{(r)}, b^{(r)}, c^{(r)}\right) = \Phi\Big(q, \tau_q, \mathcal{X}^{(r)}, \mathcal{S}^{(r-1)}, \mathcal{H}^{(r)}\Big). \tag{5}$$

The procedure incorporates newly supported information into the state, uses evidence timestamps and τ_q to interpret temporal relations, and updates the draft answer and remaining gaps. If the evidence is insufficient, it generates up to K_q follow-up queries that target the unresolved gaps while retaining relevant entity, event, and temporal constraints. This gap-driven update directs subsequent retrieval toward missing information identified from intermediate findings. Retrieved evidence is merged into the pool before the next state update.

To keep reflection focused as evidence accumulates, the controller may filter an oversized evidence view and block items judged irrelevant. Iterative retrieval terminates when the evidence is sufficient, when no follow-up query or new evidence is available, or when the maximum number of rounds R is reached. The complete iterative procedure is provided in Appendix A.1.

Final evidence and answer generation. After retrieval terminates, LGMem selects up to K_f top-ranked retained evidence items and normalizes relative time expressions in the selected evidence using their session timestamps. Let $\mathcal{X}_q$ denote the resulting final evidence context and $\mathcal{S}_q$ the terminal query state. The answer-generation LLM $\mathrm{Gen}(\cdot)$ produces

$$\hat{y} = \mathrm{Gen}(q, \tau_q, \mathcal{X}_q, \mathcal{S}_q). \tag{6}$$

The raw dialogue evidence in $\mathcal{X}_q$ grounds the answer, and $\mathcal{S}_q$ provides the query-specific findings accumulated across retrieval rounds.

3 EXPERIMENTS

3.1 EXPERIMENTAL SETUP

Datasets. We evaluate on LongMemEval-S (Wu et al., 2025) (500 questions) and BEAM (Tavakoli et al., 2026) at three dialogue-history scales: 100K tokens (400 questions), 500K tokens (700 questions), and 1M tokens (700 questions). Dataset and metric details are in Appendix C.1.

Baselines. We compare LGMem with seven representative baselines from four categories—raw-text retrieval: Dense RAG (Lewis et al., 2020); hierarchical retrieval: RAPTOR (Sarthi et al., 2024); compression-based memory: LightMem (Fang et al., 2026) and SimpleMem (Liu et al., 2026); and graph-based memory and indexing: LinearRAG (Zhuang et al., 2026), MAGMA (Jiang et al., 2026), and MemGAS (Xu et al., 2026b). Detailed baseline configurations are provided in Appendix D.2.

Table 1: LongMemEval-S performance (%) and generative memory-processing costs with GPT-5.6-Luna and Qwen3.6-27B-FP8. Within each generator, best values are bold and second-best values are underlined; 0 denotes no generative calls in that stage, and – indicates exclusion from resource comparisons.

Method	Performance (%) ↑		Build Tokens (M) ↓		Reasoning Tokens (M) ↓		Total (M) ↓	Calls ↓
	Judge	F1	In	Out	In	Out		
GPT-5.6-Luna								
Dense RAG	63.00	<u>45.63</u>	–	–	–	–	–	–
LinearRAG	57.40	40.26	–	–	–	–	–	–
RAPTOR	56.60	40.77	53.124	<u>1.067</u>	**0**	**0**	54.191	<u>4,174</u>
MAGMA	62.60	43.56	69.055	44.046	**0**	**0**	113.101	224,019
LightMem	<u>74.60</u>	20.42	<u>14.020</u>	12.342	**0**	**0**	<u>26.362</u>	10,718
SimpleMem	71.40	45.54	61.636	17.715	<u>4.324</u>	0.724	84.399	9,782
MemGAS	69.60	28.52	82.671	5.141	4.495	<u>0.051</u>	92.358	38,322
LGMem (Ours)	**80.60**	**51.97**	**0**	**0**	5.590	0.789	**6.379**	**906**
Qwen3.6-27B-FP8								
Dense RAG	57.00	<u>44.12</u>	–	–	–	–	–	–
LinearRAG	50.00	39.72	–	–	–	–	–	–
RAPTOR	52.80	40.67	54.269	<u>1.061</u>	**0**	**0**	55.329	<u>4,161</u>
MAGMA	50.80	40.08	81.163	38.294	**0**	**0**	119.457	247,441
LightMem	<u>74.40</u>	8.68	<u>17.089</u>	9.341	**0**	**0**	<u>26.429</u>	12,305
SimpleMem	58.20	43.79	66.364	12.356	<u>3.801</u>	0.393	82.914	9,623
MemGAS	48.80	15.87	86.873	2.935	4.511	<u>0.090</u>	94.409	38,322
LGMem (Ours)	**77.20**	**60.08**	**0**	**0**	5.738	0.480	**6.218**	**890**

Experimental protocol. We use GPT-5.6-Luna for generative stages and MiniLM for encoding, except that MemGAS retains its official Contriever retriever. We also evaluate all methods with Qwen3.6-27B-FP8 in non-thinking mode on LongMemEval-S and BEAM-100K. Following the official protocols, GPT-4o-mini evaluates LongMemEval-S and GPT-4.1-mini evaluates BEAM with dialogue-level macro-averaging; methods within each BEAM scale share an answer prompt. Judge score is primary and token F1 diagnostic. Costs cover generative memory construction and query reasoning, excluding final answer generation and evaluation.

3.2 Main Results

LongMemEval-S. As shown in Table 1, with GPT-5.6-Luna, LGMem achieves a Judge score of 80.60, outperforming the strongest baseline, LightMem, by 6.00 percentage points while reducing total generative token consumption by 75.80%. With Qwen3.6-27B-FP8, LGMem achieves a Judge score of 77.20, maintaining a 2.80-point advantage over LightMem while reducing token consumption by 76.47%. Across both settings, LGMem obtains the highest token F1 and the lowest total generative token consumption among the methods included in the resource comparison. LGMem also exhibits a favorable quality–runtime trade-off under local Qwen serving; detailed results and analysis are provided in Appendix E.6.

BEAM. As reported in Table 2, LGMem achieves the highest Judge score at all three dialogue-history scales. It surpasses the strongest baseline by 6.01 percentage points on BEAM-100K, with the margin increasing to 8.52 and 9.70 points on BEAM-500K and BEAM-1M. Compared with these strongest baselines, LGMem reduces total generative token consumption by 84.49%, 66.57%, and 82.16% at the three scales, respectively. Moreover, as the dialogue history grows tenfold from 100K to 1M tokens, its per-question token consumption increases only from 16.00K to 19.29K, indicating that query-time processing does not grow proportionally with the full dialogue history. LightMem constructs its persistent memory entries from user turns only, which comprise 5.4–8.1% of source-dialogue tokens across the three scales (Appendix E.3); it has the lowest generative memory-processing cost, but its Judge scores remain 10.17–15.78 points below LGMem.

Table 2: BEAM performance (%) and generative memory-processing costs with GPT-5.6-Luna across 100K–1M-token histories. Qwen3.6-27B-FP8 results on BEAM-100K are reported in Appendix E.2. Within each scale, best values are bold and second-best values are underlined; 0 denotes no generative calls in that stage, and – indicates exclusion from resource comparisons.

Method	Performance (%) ↑		Build Tokens (M) ↓		Reasoning Tokens (M) ↓		Total (M) ↓	Calls ↓
	Judge	F1	In	Out	In	Out		
BEAM-100K								
Dense RAG	54.25	29.05	–	–	–	–	–	–
LinearRAG	56.26	29.56	–	–	–	–	–	–
RAPTOR	47.07	27.67	4.339	1.298	**0**	**0**	5.636	5,306
MAGMA	56.76	28.66	2.819	1.155	**0**	**0**	3.974	5,094
LightMem	51.34	29.73	0.445	0.405	**0**	**0**	**0.850**	**368**
MemGAS	61.10	29.58	4.636	0.035	36.277	0.308	41.256	567
SimpleMem	55.61	28.88	3.424	1.254	0.430	0.329	5.437	1,620
LGMem (Ours)	**67.12**	**30.26**	**0**	**0**	5.449	0.950	6.399	891
BEAM-500K								
Dense RAG	52.06	27.50	–	–	–	–	–	–
LinearRAG	53.07	27.78	–	–	–	–	–	–
RAPTOR	48.46	27.86	25.302	5.315	**0**	**0**	30.617	21,033
MAGMA	54.77	28.74	21.511	8.189	**0**	**0**	29.700	38,055
LightMem	54.65	**31.48**	2.755	2.261	**0**	**0**	**5.017**	2,258
MemGAS	51.19	26.75	35.515	0.144	80.856	0.342	116.856	**1,400**
SimpleMem	56.30	28.88	26.330	9.524	0.757	0.573	37.184	5,194
LGMem (Ours)	**64.82**	30.04	**0**	**0**	10.651	1.781	12.432	1,731
BEAM-1M								
Dense RAG	50.87	28.93	–	–	–	–	–	–
LinearRAG	53.00	28.60	–	–	–	–	–	–
RAPTOR	47.27	28.97	43.085	6.230	**0**	**0**	49.315	24,570
MAGMA	52.34	29.49	40.524	14.738	**0**	**0**	55.262	69,169
LightMem	53.90	**32.00**	4.245	3.300	**0**	**0**	**7.545**	3,274
MemGAS	52.41	26.84	64.861	0.145	80.538	0.394	145.938	**1,372**
SimpleMem	55.64	28.62	54.071	20.200	0.779	0.617	75.667	8,369
LGMem (Ours)	**65.34**	30.57	**0**	**0**	11.552	1.948	13.500	1,874

Overall analysis. Across all evaluated settings, LGMem achieves the highest Judge score while using substantially fewer generative tokens than the strongest QA baseline. Paired bootstrap analysis confirms that these improvements are statistically significant in all four primary settings (Appendix E.4). Compression-based methods, particularly LightMem and SimpleMem, are generally the strongest baselines, suggesting that compression can improve information density and reduce retrieval noise. However, LightMem's low token F1 and the greater generator sensitivity of SimpleMem indicate potential losses in lexical detail and robustness. Dense RAG, LinearRAG, and RAPTOR are less competitive overall, while graph-based methods show mixed results: MemGAS performs strongly on BEAM-100K but incurs high query-time cost. These patterns suggest that neither compression nor structured indexing alone guarantees sufficient evidence for every query.

3.3 Ablation Study

We evaluate five ablated variants with GPT-5.6-Luna on all 500 LongMemEval-S questions. The comparison assesses graph retrieval, TextRank keyword anchors, iterative reflection, temporal processing, and the joint removal of graph retrieval and reflection. Figure 3 reports the module ablations, while Figure 4 examines the effect of the maximum reflection depth.

Graph memory and anchors. Replacing graph retrieval with dense segment retrieval reduces the score from 80.60% to 76.40%, showing the value of structured graph access over direct similar-

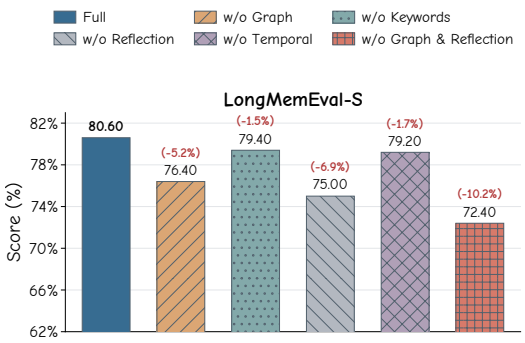


Figure 3: Module ablations on LongMemEval-S with GPT-5.6-Luna.

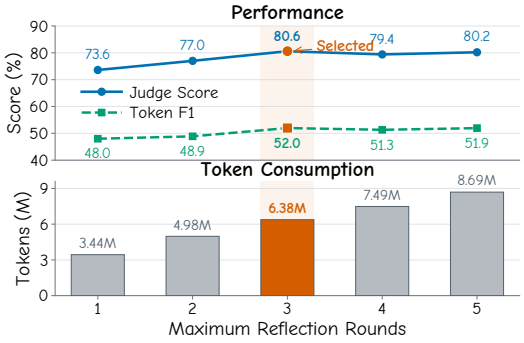


Figure 4: Score, F1, and tokens by reflection depth on LongMemEval-S.

ity matching. Removing TextRank keyword anchors lowers the score to 79.40%, indicating that keyword anchors complement entity anchors.

Query-time access. Removing reflection reduces the score from 80.60% to 75.00%, indicating that evidence assessment and targeted query refinement are central to query-time access. Removing both graph retrieval and reflection lowers the score further to 72.40%, showing that structured retrieval and iterative evidence completion both contribute to the full system. Disabling temporal processing yields a score of 79.20%, indicating that temporal cues also support performance.

Reflection depth. Increasing the maximum reflection rounds from one to three improves the Judge score from 73.60% to 80.60% and Token F1 from 47.99% to 51.97%, while increasing the corresponding token consumption from 3.44M to 6.38M (Figure 4). Further rounds provide no accuracy gain: at four and five rounds, the Judge scores are 79.40% and 80.20%, respectively, while token consumption rises to 7.49M and 8.69M. The default three-round setting attains the highest Judge score and Token F1, while using fewer tokens than four or five rounds.

3.4 Case Study

Figure 5 presents a case in which LightMem misses the answer-bearing event and SimpleMem ranks it below a distractor. Starting from similarly incomplete evidence, LGMem identifies the temporal gap and recovers the event with a targeted second-round query. The recovered evidence supports the correct answer, illustrating how stateful reflection improves evidence completeness when initial retrieval is insufficient. Appendix E.7 provides a complementary case of initial cross-session evidence coverage.

4 Related Work

Long-term memory management and compression. Long-term memory research supports sustained agent interaction through experience retention, continuous updating, and selective preservation (Park et al., 2023; Zhong et al., 2024). For storage management, MemGPT (Packer et al., 2023), MemoryOS (Kang et al., 2025), and MemOS (Li et al., 2025) manage limited contexts and external memory through tiered storage or unified scheduling. For content organization, LightMem (Fang et al., 2026) uses topic-based summaries and offline consolidation, SimpleMem (Liu et al., 2026) improves information density through semantic compression, SeCom (Pan et al., 2025) combines dialogue segmentation and compression-based denoising to improve retrieval, and MEM1 (Zhou et al., 2026) learns a compact state for both memory and reasoning. By contrast, LGMem retains raw dialogue evidence rather than replacing it with summaries or compact states. It organizes this evidence through non-generative incremental graph construction and gathers query-specific evidence through reflection-guided iterative access.

Structured memory and graph indexing. Structured memory connects dispersed information through relations and hierarchies. The graph variant of Mem0 (Chhikara et al., 2025) and A-

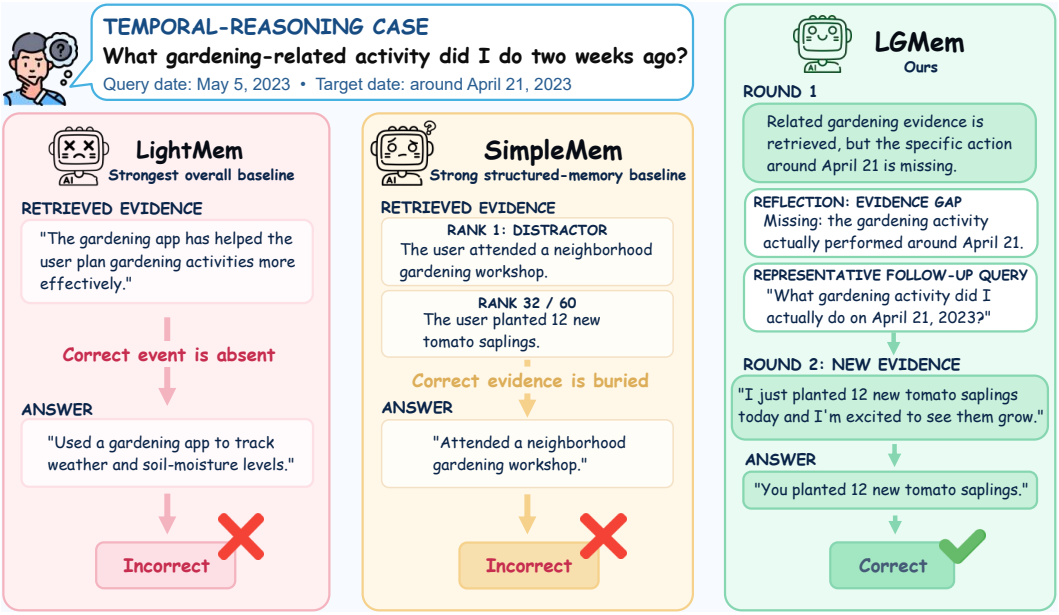


Figure 5: A case study of reflection-guided retrieval on LongMemEval-S. LGMem recovers an answer-bearing event missed or under-ranked by one-shot baselines through a targeted second-round query.

MEM (Xu et al., 2025) organize memory through explicit relations or dynamic links; Zep (Rasmussen et al., 2025) and APEX-MEM (Banerjee et al., 2026) further address temporal evolution. Multi-view relations and higher-order associations expand graph memory representations (Jiang et al., 2026; Yue et al., 2026), while MemGAS (Xu et al., 2026b) and StructMem (Xu et al., 2026a) strengthen evidence associations through multi-granularity or hierarchical structures. Related graph retrieval research also uses structural propagation for associative access (Gutiérrez et al., 2025). LinearRAG (Zhuang et al., 2026) organizes entities, sentences, and segments as two relation-free bipartite graphs and ranks segments using personalized PageRank. Inspired by this relation-free indexing design, LGMem connects anchors, sentences, and segments without explicit relation extraction. Unlike LinearRAG, LGMem builds a unified three-layer graph with both entity and keyword anchors and retrieves evidence through query-activated local multi-hop propagation and segment-level score aggregation rather than personalized PageRank.

Query-driven retrieval and reflective access. Query-driven access adapts memory selection to information needs. Mnemis (Tang et al., 2026) supplements similarity retrieval with hierarchical selection; IRCoT (Trivedi et al., 2023) and Self-RAG (Asai et al., 2024) combine retrieval with chain-of-thought reasoning and evaluation of evidence and generated content, respectively. For long-term memory, MRAgent (Ji et al., 2026) and REMem (Shu et al., 2026) collect evidence through iterative graph exploration and retrieval, while Hindsight (Latimer et al., 2026) combines retrieval and reflection through memory operations. By comparison, LGMem decouples graph retrieval from reflection: lightweight local propagation gathers evidence, while stateful reflection maintains an explicit query state of supported facts, temporal relations, and unresolved gaps across rounds. This state drives targeted follow-up queries without requiring LLM-controlled graph traversal.

5 Conclusion

We introduce LGMem, a long-term dialogue memory framework that combines lightweight incremental construction with reflection-guided iterative retrieval. LGMem retains raw dialogue evidence without LLM calls during memory construction and uses stateful reflection to identify evidence gaps and guide iterative retrieval across sessions at query time. Experiments on LongMemEval-S and BEAM show that LGMem achieves better overall answer quality than existing baselines and a competitive trade-off between answer quality and generative memory-processing costs.

AI USE STATEMENT

During the preparation of this manuscript, the authors used large language models (LLMs) for language polishing, including wording refinement and improvements to clarity and readability. LLMs were not used to generate research ideas, develop the methodology or experimental design, produce the reported results, or draw scientific conclusions. The authors have reviewed and edited all such content and take full responsibility for the final manuscript.

ETHICS STATEMENT

Our experiments use existing LongMemEval-S and BEAM benchmarks and involve no collection of new user data. Long-term memory systems may retain sensitive dialogue content, requiring careful attention to privacy and data protection in real-world deployment.

REPRODUCIBILITY STATEMENT

Appendices A–F provide the stateful access algorithm, key parameters, experimental configurations, evaluation metrics, and complete prompt templates. Code, released configurations, data-preparation and evaluation scripts, and reference aggregate results are available in our anonymous repository at `https://anonymous.4open.science/r/lgmem-anonymous-16B1`.

REFERENCES

Akari Asai, Zeqiu Wu, Yizhong Wang, Avi Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *International Conference on Learning Representations*, pp. 9112–9141, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/25f7be9694d7b32d5cc670927b8091e1-Abstract-Conference.html.

Pratyay Banerjee, Masud Moshtaghi, Shivashankar Subramanian, Amita Misra, and Ankit Chadha. APEX-MEM: Agentic semi-structured memory with temporal reasoning for long-term conversational AI. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16470–16489, 2026. doi: 10.18653/v1/2026.acl-long.749.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. In *ECAI 2025*, pp. 2993–3000. IOS Press, 2025. doi: 10.3233/FAIA251160. URL https://doi.org/10.3233/FAIA251160.

Yiming Du, Wenyu Huang, Danna Zheng, Zhaowei Wang, Sebastien Montella, Mirella Lapata, Kam-Fai Wong, and Jeff Z. Pan. Rethinking memory in LLM based agents: Representations, operations, and emerging topics. *arXiv preprint arXiv:2505.00675*, 2025. doi: 10.48550/arXiv.2505.00675. URL https://arxiv.org/abs/2505.00675.

Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, Huajun Chen, and Ningyu Zhang. LightMem: Lightweight and efficient memory-augmented generation. In *International Conference on Learning Representations*, pp. 98706–98729, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/a05b72653ec5b473732129829ae04195-Abstract-Conference.html.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. From RAG to memory: Non-parametric continual learning for large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 21497–21515, 2025. URL https://proceedings.mlr.press/v267/gutierrez25a.html.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength natural language processing in python, 2020.

Shuo Ji, Yibo Li, and Bryan Hooi. Memory is reconstructed, not retrieved: Graph memory for LLM agents. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, 2026. URL https://icml.cc/virtual/2026/poster/60697.

Dongming Jiang, Yi Li, Guanpeng Li, and Bingzhe Li. MAGMA: A multi-graph based agentic memory architecture for AI agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 36848–36865, 2026. doi: 10.18653/v1/2026.acl-long.1709.

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. 25961–25970. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.1318. URL https://aclanthology.org/2025.emnlp-main.1318/.

Christopher Latimer, Nicolò Boschi, Andrew Neeser, Chris Bartholomew, Gaurav Srivastava, Xuan Wang, and Naren Ramakrishnan. Hindsight: Structured agent memory that retains, recalls, and reflects. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 275–285, 2026. doi: 10.18653/v1/2026.acl-demo.27.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Zhiyu Li, Chenyang Xi, Chunyu Li, Ding Chen, Boyu Chen, Shichao Song, Simin Niu, Hanyu Wang, Jiawei Yang, Chen Tang, Qingchen Yu, Jihao Zhao, Yezhaohui Wang, Peng Liu, Zehao Lin, Pengyuan Wang, Jiahao Huo, Tianyi Chen, Kai Chen, Kehang Li, Zhen Tao, Huayi Lai, Hao Wu, Bo Tang, Zhengren Wang, Zhaoxin Fan, Ningyu Zhang, Linfeng Zhang, Junchi Yan, Mingchuan Yang, Tong Xu, Wei Xu, Huajun Chen, Haofen Wang, Hongkang Yang, Wentao Zhang, Zhi-Qin John Xu, Siheng Chen, and Feiyu Xiong. MemOS: A memory OS for AI system. *arXiv preprint arXiv:2507.03724*, 2025. URL https://arxiv.org/abs/2507.03724.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. SimpleMem: Efficient lifelong memory for LLM agents. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, 2026. URL https://icml.cc/virtual/2026/poster/61640.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638.

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. doi: 10.18653/v1/2024.acl-long.747.

Rada Mihalcea and Paul Tarau. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pp. 404–411, 2004. URL https://aclanthology.org/W04-3252/.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023. URL https://arxiv.org/abs/2310.08560.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Xufang Luo, Hao Cheng, Dongsheng Li, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Jianfeng Gao. SeCom: On memory construction and retrieval for personalized conversational agents. In *International Conference on Learning Representations*, pp. 91851–91885, 2025.

URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/e56f394bbd4f0ec81393d767caa5a31b-Abstract-Conference.html.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pp. 1–22, 2023. doi: 10.1145/3586183.3606763.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. Zep: A temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956*, 2025. URL https://arxiv.org/abs/2501.13956.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 3982–3992, 2019. doi: 10.18653/v1/D19-1410.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. RAPTOR: Recursive abstractive processing for tree-organized retrieval. In *International Conference on Learning Representations*, pp. 32628–32649, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/8a2acd174940dbca361a6398a4f9df91-Abstract-Conference.html.

Yiheng Shu, Saisri Padmaja Jonnalagadda, Xiang Gao, Bernal Jiménez Gutiérrez, Weijian Qi, Kamalika Das, Huan Sun, and Yu Su. REMem: Reasoning with episodic memory in language agent. In *International Conference on Learning Representations*, pp. 55958–55987, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/5b93eda3b396a7f665a26fbf53655adc-Abstract-Conference.html.

Zihao Tang, Xin Yu, Ziyu Xiao, Zengxuan Wen, Zelin Li, Jiaxi Zhou, Hualei Wang, Haohua Wang, Haizhen Huang, Weiwei Deng, Feng Sun, and Qi Zhang. Mnemis: Dual-route retrieval on hierarchical graphs for long-term LLM memory. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23914–23928, 2026. doi: 10.18653/v1/2026.acl-long.1096.

Mohammad Tavakoli, Alireza Salemi, Carrie Ye, Mohamed Abdalla, Hamed Zamani, and J. Ross Mitchell. Beyond a million tokens: Benchmarking and enhancing long-term memory in LLMs. In *International Conference on Learning Representations*, pp. 133343–133429, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/d7f0cfa0fe759b033d5262e1bb7d4065-Abstract-Conference.html.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10014–10037, 2023. doi: 10.18653/v1/2023.acl-long.557.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *International Conference on Learning Representations*, pp. 86809–86836, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/d813d324dbf0598bbdc9c8e79740ed01-Abstract-Conference.html.

Buqiang Xu, Yijun Chen, Jizhan Fang, Ruobin Zhong, Yunzhi Yao, Yuqi Zhu, Lun Du, and Shumin Deng. StructMem: Structured memory for long-horizon behavior in LLMs. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 122–146, 2026a. doi: 10.18653/v1/2026.acl-short.12.

Derong Xu, Yi Wen, Pengyue Jia, Yingyi Zhang, Wenlin Zhang, Yichao Wang, Huifeng Guo, Ruiming Tang, Xiangyu Zhao, Enhong Chen, and Tong Xu. From single to multi-granularity: Toward long-term memory association and selection of conversational agents. In *International Conference on Learning Representations*, pp. 91581–91617, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/93e444b4a1a64815631c3c74977ee200-Abstract-Conference.html.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-Mem: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems*, volume 38, pp. 17577–17604, 2025. doi: 10.52202/085713-0593.

Juwei Yue, Chuanrui Hu, Jiawei Sheng, Zuyi Zhou, Wenyuan Zhang, Tingwen Liu, Li Guo, and Yafeng Deng. HyperMem: Hypergraph memory for long-term conversations. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 35237–35254, 2026. doi: 10.18653/v1/2026.acl-long.1627.

Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model-based agents. *ACM Transactions on Information Systems*, 43(6):1–47, 2025. doi: 10.1145/3748302. URL https://doi.org/10.1145/3748302.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. MemoryBank: Enhancing large language models with long-term memory. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19724–19731, 2024. doi: 10.1609/aaai.v38i17.29946.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Bryan Kian Hsiang Low, and Paul Liang. MEM1: Learning to synergize memory and reasoning for efficient long-horizon agents. In *International Conference on Learning Representations*, pp. 58413–58438, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/5fc8b3bdfbb9167b5144df5d3fae4616-Abstract-Conference.html.

Luyao Zhuang, Shengyuan Chen, Yilin Xiao, Huachi Zhou, Yujing Zhang, Hao Chen, Qinggang Zhang, and Xiao Huang. LinearRAG: Linear graph retrieval augmented generation on large-scale corpora. In *International Conference on Learning Representations*, pp. 147053–147075, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/ee1955739b91db042e659b6f782a5e79-Abstract-Conference.html.

A Algorithms and State Management

A.1 Stateful Access Procedure

Algorithm 1 details LGMem's stateful access and answer-generation procedure. Scoring and state-update rules are defined in Appendix A.2.

Algorithm 1 Bounded stateful graph access and answer generation.

Input: question q with time reference τ_q, graph $\mathcal{G}$, evidence limits K, K_f, filter threshold T_f, round/query limits R, K_q, context limits C_r, C_a, and confidence thresholds θ_c, θ_i
Output: answer $\hat{y}$

Initialize $\mathcal{P} \leftarrow \emptyset$, $\mathcal{B} \leftarrow \emptyset$, $\mathcal{S} \leftarrow \emptyset$, and $\mathcal{H} \leftarrow [\,]$; set $\mathcal{Q}^{(1)} \leftarrow \{q\}$ and $\mathcal{I}_{\text{keep}} \leftarrow \perp$.
for round $r = 1, \ldots, R$ **do**
  For each $q' \in \mathcal{Q}^{(r)}$, retrieve K segments and their supporting sentences; merge unblocked items into $\mathcal{P}$ by stable identifier and record repeated retrievals.
  Append $\mathcal{Q}^{(r)}$ to $\mathcal{H}$ and rank $\mathcal{P}$ by best rank and the repeated-hit bonus.
  Construct a numbered evidence view $\mathcal{X}^{(r)}$ subject to the reflection context limit C_r.
  if $r > 1$ and $|\mathcal{P}| > T_f$ **then**
    $\mathcal{I}_{\text{keep}} \leftarrow \textsc{Filter}(q, \mathcal{X}^{(r)}, K_f)$, with initially top-ranked and repeatedly retrieved items protected.
  end if
  $\mathcal{R}_r \leftarrow \textsc{Reflect}(q, \tau_q, \mathcal{X}^{(r)}, \mathcal{S}, \mathcal{H})$.
  Add items judged irrelevant with confidence at least θ_i to $\mathcal{B}$, remove them from $\mathcal{P}$, and merge $\mathcal{R}_r$ into $\mathcal{S}$.
  if $\mathcal{R}_r$.*can_answer* and $\mathcal{R}_r$.*confidence* $\geq \theta_c$ **then break**.
  Let $\mathcal{Q}^{(r+1)}$ contain at most K_q normalized, nonempty subqueries from $\mathcal{R}_r$ that do not occur in $\mathcal{H}$.
  if $\mathcal{Q}^{(r+1)} = \emptyset$, or $r > 1$ and this round added no new evidence, **then break**.
end for
Select up to K_f ranked items from $\mathcal{P}$, restricted to $\mathcal{I}_{\text{keep}}$ when it is not $\perp$, to form $\mathcal{X}_q^{\text{raw}}$ subject to C_a; set $\mathcal{S}_q \leftarrow \mathcal{S}$.
Time-normalize $\mathcal{X}_q^{\text{raw}}$ using each evidence timestamp to obtain $\mathcal{X}_q$; **return** $\hat{y} \leftarrow \text{Gen}(q, \tau_q, \mathcal{X}_q, \mathcal{S}_q; C_a)$.

A.2 Scoring, Evidence Fusion, and Validation

Segment ranking. The graph evidence score $g(d_k)$ is the sum of at most the top K_g supporting-sentence scores in the segment divided by $\log(2 + |d_k|_{\text{tok}})$, where $|d_k|_{\text{tok}}$ is the segment's token length. For each retrieval query, graph evidence scores and dense similarities are normalized separately. For scores x_k of either type, we use

$$\widetilde{x}_k = \frac{x_k - \min_j x_j}{\max_j x_j - \min_j x_j}.$$

If all scores of a type are equal, their normalized values are set to 1; a segment without a graph evidence score receives 0 for that term. The dialogue-recency score $\rho(d_k)$ linearly maps segment order in the dialogue history into $[0, 1]$: the earliest segment receives 0, the latest 1, and a single segment 0. The graph evidence coefficient is fixed at 1, and auxiliary coefficients follow Table 3.

Cross-round evidence fusion. Each pool item records a stable identifier, raw text, best rank across queries $r_{\min}(e)$, and hit count $n_{\text{hit}}(e)$; identifiers are segment or sentence IDs. After duplicate evidence is merged by identifier, its fusion score is

$$\psi(e) = \frac{1}{r_{\min}(e)} + \beta \log n_{\text{hit}}(e), \tag{7}$$

Here, $\beta \geq 0$ controls the logarithmic repeated-hit bonus. Both reflection views and final evidence selection use this ranking.

Filtering changes only the evidence view presented to the controller and does not delete items from the accumulated pool. The retained identifiers in $\mathcal{I}_{\text{keep}}$ constrain final evidence selection, whereas evidence blocked as irrelevant is removed from subsequent retrieval.

Cross-round state merging. The system merges facts retained from previous rounds with newly extracted facts and deduplicates them by text. Structured records are deduplicated by their date, subject, and value fields. Nonempty draft-answer and evidence-gap fields from the current round replace their previous values; otherwise, the previous values are retained. Newly blocked identifiers accumulate as $\mathcal{B}^{(r)} = \mathcal{B}^{(r-1)} \cup \Delta\mathcal{B}^{(r)}$.

Output validation. Reflection prompts number current evidence as $[1], [2], \ldots$. After parsing and index validation, the current-round mapping converts these indices to stable identifiers for blocking, reordering, and evidence references in the state. Repeated subqueries are removed before the next retrieval round.

B Implementation Settings and Complexity

B.1 Key Configurations

We use a single shared set of graph-retrieval and controller settings across all benchmarks (Table 3). Figure 4 reports the effect of varying the maximum reflection depth.

Table 3: Key LGMem implementation settings.

Component	Setting
Anchor extraction	spaCy `en_core_web_trf` NER and sentence-level TextRank.
Embedding	`all-MiniLM-L6-v2` (Reimers & Gurevych, 2019).
Anchor activation	At most 16 query concepts and 32 semantic seeds; cosine threshold 0.4; top-3 dense sentence seeds when anchor matching is empty.
Propagation	2 hops; at most 3 sentences per anchor; top-3 sentence aggregation with logarithmic length penalty.
Segment ranking	Graph/dense/dialogue-recency weights 1.0/0.15/0.05.
Reflection	At most 3 rounds and 3 new subqueries per round; answer-confidence threshold 0.75; irrelevant-block threshold 0.85.
Evidence pool	Preserve initial top-5; repeated-hit bonus 0.15; append up to 5 supporting-sentence items per base retrieval.
Generation	Temperature 0 for reflection, filtering and final answers.

Table 4 reports candidate segment counts K, final evidence counts K_f, and reflection and answer limits for each benchmark. LongMemEval-S uses the same settings for GPT-5.6-Luna and Qwen3.6-27B-FP8.

Table 4: LGMem evidence and generation limits. Input and output limits are measured in characters and tokens, respectively.

Benchmark	K/K_f	Reflection	Answer
LongMemEval-S	15/25	24K/2048	32K/512
BEAM-100K/500K/1M	30/20	64K/4000	64K/1024

Here, K controls the number of segments retrieved by each query, whereas K_f caps the final evidence selected from the pool accumulated across queries and rounds; hence, K_f can exceed K. Appendix E.5 reports sensitivity to both budgets on BEAM-100K.

B.2 Complexity

Let N_a, N_s, N_d denote the numbers of anchors, sentences, and segments, respectively, M the number of edges, d the embedding dimension, and $\Delta\mathcal{V}_t$ and $\Delta\mathcal{L}_t$ the new nodes and links introduced by the incoming session. After text extraction for the new session, appending the graph structure requires $O(|\Delta\mathcal{V}_t| + |\Delta\mathcal{L}_t|)$ work; embeddings are encoded and cached on first access. Storage for raw text, provenance fields, graph structure, and all cached embeddings is $O(B_{\text{text}} + M + (N_a + N_s + N_d)d)$, where B_{text} is the size of text and provenance fields.

With cached node embeddings, worst-case global similarity computation for one base retrieval is $O((N_a + N_s + N_d)d)$. Local propagation additionally scans active-node adjacency lists and performs candidate selection and ranking; the hop and per-anchor sentence limits bound traversal breadth, while the actual scan volume depends on node degrees. Stateful access executes at most $1 + (R-1)K_q$ base queries, with at most one reflection call per round.

C Evaluation Protocol

C.1 Datasets and Metrics

LongMemEval-S. LongMemEval-S (Wu et al., 2025) contains 500 QA instances, each paired with a user–assistant history of approximately 50 sessions and 115K tokens. Its six abilities cover single-session user facts, single-session assistant information, single-session preferences, multi-session reasoning, temporal reasoning, and knowledge updates. Questions may require locating evidence across sessions, interpreting temporal references, or resolving information that changes over time. Each method receives the ordered dialogue history with speaker labels and timestamps, the question, and the question date when available. Category labels and answer-session identifiers are retained only as evaluation metadata and are not included in model prompts.

BEAM. We evaluate BEAM (Tavakoli et al., 2026) at history lengths of 100K, 500K, and 1M tokens. Each evaluated dialogue is paired with 20 questions spanning ten abilities: abstention, contradiction resolution, event ordering, information extraction, instruction following, knowledge updates, multi-session reasoning, preference following, summarization, and temporal reasoning. These settings test memory access and reasoning as the dialogue history grows while preserving a common set of abilities across scales.

Evaluation metrics. For LongMemEval-S, the primary metric is binary judge accuracy under the official yes/no correctness prompt, using GPT-4o-mini as the evaluator. For BEAM, GPT-4.1-mini applies the official category-specific protocol: event ordering uses normalized Kendall τ, while the other nine abilities use rubric scores of 0, 0.5, or 1. Scores are first averaged within each dialogue and then macro-averaged across abilities. For both datasets, token F1 measures normalized token overlap with the reference answer and is reported as a diagnostic metric rather than the primary score. Table 5 summarizes the evaluated settings.

Table 5: Dataset statistics and primary evaluation metrics.

Dataset	Evaluation size	History tokens	Abilities	Primary metric
LongMemEval-S	500 QA	∼115K	6	Judge accuracy
BEAM-100K	20 chats / 400 QA	100K	10	Official macro score
BEAM-500K	35 chats / 700 QA	500K	10	Official macro score
BEAM-1M	35 chats / 700 QA	1M	10	Official macro score

C.2 Answer Generation and Cost Accounting

Prompt 3 specifies LGMem's answer template for LongMemEval-S. For BEAM, all methods use the shared answer template in Prompt 4. We report generative tokens and model calls for memory construction and query-time reasoning, excluding final answer generation and evaluation. Because all questions in a BEAM dialogue use the same constructed memory, its construction cost is counted once per dialogue.

D Baseline Implementations

D.1 Baseline Methods

Dense RAG retrieves raw dialogue units by embedding similarity. LinearRAG (Zhuang et al., 2026) uses relation-free graph indexing over entities, sentences, and segments, while RAPTOR (Sarthi

et al., 2024) organizes clustered summaries in a retrieval tree. MAGMA (Jiang et al., 2026) constructs a temporal relational graph over dialogue messages. LightMem (Fang et al., 2026) combines topic segmentation, compression, and offline consolidation; SimpleMem (Liu et al., 2026) uses semantically compressed memory with adaptive multi-view retrieval; and MemGAS (Xu et al., 2026b) associates multiple memory granularities in a graph and filters retrieved evidence.

D.2 Retrieval and Answer Configurations

Table 6 reports the key parameters that determine each baseline's outputs. Parameter selection follows the original paper and official code whenever their settings apply to the target dataset. If no target-dataset configuration is available, we preserve the published memory and retrieval procedure and introduce only the input-unit or evaluation-interface adaptation required by the benchmark. MiniLM is shared across baselines to control the encoder, except that MemGAS retains its official Contriever to avoid changing its retrieval pipeline; final-answer templates are shared within each dataset. All BEAM scales use the same frozen settings and differ only in history length and evaluation instances. RAPTOR has no official LongMemEval-S preprocessing, so we follow the MemGAS LongMemEval comparison by treating each session as a leaf and apply the official RAPTOR tree construction above it; BEAM retains the official 100-token leaves. LongMemEval-S uses GPT-5.6-Luna and Qwen3.6-27B-FP8; the main BEAM comparison uses GPT-5.6-Luna, with the BEAM-100K Qwen results reported in Appendix E.2.

Table 6: Key baseline configurations and adaptation rationale.

Method / dataset	**Key configuration and source**
Dense RAG / LongMemEval-S (Lewis et al., 2020)	Retrieval-only control over raw messages from both roles; MiniLM cosine top-10; 32K-character answer context; source identifiers hidden.
LinearRAG / LongMemEval-S (Zhuang et al., 2026)	Official graph pipeline with MiniLM replacing the upstream encoder; top-5 evidence; 32K-character answer context.
LinearRAG / BEAM (Zhuang et al., 2026)	Official graph pipeline over user–assistant dialogue pairs; MiniLM top-5 evidence; 64K-character answer context.
RAPTOR / LongMemEval-S (Sarthi et al., 2024)	Official tree construction applied to session-level leaves following the MemGAS LongMemEval treatment; 256-token cluster summaries; MiniLM top-6 collapsed-tree retrieval; 16K-token answer context.
MAGMA / LongMemEval-S (Jiang et al., 2026)	Fair adaptation of its message-level temporal graph; question-only routing and MiniLM top-20; oracle fields excluded.
Dense RAG / BEAM	Dataset-native retrieval-only control over raw user–assistant pairs; MiniLM cosine top-5; 64K-character answer context.
RAPTOR / BEAM (Sarthi et al., 2024)	Official 100-token leaves and 256-token cluster summaries; MiniLM top-6 collapsed-tree retrieval with a 16K-token retrieval budget; 64K-character shared answer context.
MAGMA / BEAM (Jiang et al., 2026)	Per-chat message graph; question-only routing; MiniLM top-20; 4,200-token evidence budget; oracle fields excluded.
LightMem / LongMemEval-S (Fang et al., 2026)	Official memory pipeline and repository settings, including LLMLingua2 preprocessing.
LightMem / BEAM	Cross-dataset transfer preserving the official user-only persistent-memory extraction, LLMLingua2, topic segmentation, and offline consolidation; MiniLM top-20; one memory per chat shared by its questions.
SimpleMem / LongMemEval-S (Liu et al., 2026)	Paper-aligned 40-turn windows with overlap 1; semantic/keyword/structured retrieval budgets 25/5/5; MiniLM.
SimpleMem / BEAM	Paper-aligned transfer using 20-turn windows with overlap 5 and adaptive retrieval of 3–20 memories; MiniLM.
MemGAS (Xu et al., 2026b)	Official multi-granularity pipeline and Contriever top-3 retrieval; the BEAM transfer uses one graph per chat and adapts only the evidence-filtering prompt.

E Additional Results and Analysis

E.1 Category-Level Results

Table 7 reports category-level results on LongMemEval-S and BEAM-100K. For LongMemEval-S, Tmp., Mul., KU, Sgl-U, Sgl-A, and Pref. denote temporal reasoning, multi-session reasoning, knowledge updates, single-session user facts, single-session assistant information, and single-session preferences. For BEAM, ABS, CR, EO, IE, IF, KU, MSR, PF, SUM, and TR denote abstention, contradiction resolution, event ordering, information extraction, instruction following, knowledge updates, multi-session reasoning, preference following, summarization, and temporal reasoning. Avg. is the official overall score reported in Tables 1 and 2.

Table 7: Category-level QA scores (%) on LongMemEval-S (Judge accuracy) and BEAM-100K (official macro score) with GPT-5.6-Luna. Best results within each column are bold; second-best results are underlined.

(a) LongMemEval-S

Method	Tmp.	Mul.	KU	Sgl-U	Sgl-A	Pref.	Avg.
Dense RAG	51.13	50.38	71.79	90.00	92.86	30.00	63.00
LinearRAG	48.87	35.34	70.51	82.86	92.86	33.33	57.40
RAPTOR	36.09	47.37	73.08	78.57	94.64	23.33	56.60
MAGMA	61.65	41.35	85.90	81.43	89.29	6.67	62.60
LightMem	**84.96**	75.94	88.46	92.86	33.93	20.00	74.60
SimpleMem	60.15	**80.45**	79.49	92.86	50.00	**50.00**	71.40
MemGAS	73.68	54.89	64.10	87.14	**96.43**	40.00	69.60
LGMem (Ours)	79.70	72.18	**92.31**	**94.29**	94.64	33.33	**80.60**

(b) BEAM-100K

Method	ABS	CR	EO	IE	IF	KU	MSR	PF	SUM	TR	Avg.
Dense RAG	**85.00**	54.69	27.60	64.01	52.50	45.00	**63.57**	81.25	31.43	37.50	54.25
LinearRAG	80.00	59.38	39.48	73.96	56.88	36.25	55.57	86.88	37.92	36.25	56.26
RAPTOR	72.50	56.88	28.06	53.80	47.50	36.25	37.27	78.75	21.61	38.12	47.07
MAGMA	77.50	60.31	40.66	61.88	51.25	43.75	48.90	**88.12**	40.86	54.37	56.76
LightMem	83.75	64.38	37.61	51.25	35.62	42.50	48.55	69.38	19.75	60.62	51.34
MemGAS	72.50	66.88	36.60	71.88	58.75	**57.50**	58.49	78.75	**59.70**	50.00	61.10
SimpleMem	72.50	60.00	34.83	71.98	55.00	41.25	55.83	72.50	33.45	58.75	55.61
LGMem (Ours)	76.25	**75.62**	**44.97**	**76.98**	**64.38**	**57.50**	61.38	85.00	50.95	**78.12**	**67.12**

On BEAM-100K, LGMem performs best on contradiction resolution, event ordering, and temporal reasoning, and ties MemGAS on knowledge updates. It exceeds the second-best methods by 8.74 and 17.50 percentage points on contradiction resolution and temporal reasoning, respectively. These results show that LGMem is particularly effective for queries that require resolving conflicting or temporally distributed evidence. LGMem ranks second on multi-session reasoning, where global evidence integration remains an area for improvement.

E.2 BEAM-100K Results with Qwen

We further evaluate all methods on BEAM-100K with Qwen3.6-27B-FP8 in non-thinking mode. All methods use the same shared BEAM answer prompt and the official GPT-4.1-mini evaluation protocol. Resource accounting follows the main comparison: token costs and calls include generative memory construction and query-time reasoning, but exclude final answer generation and evaluation. Table 8 reports the results.

Table 8: BEAM-100K performance (%) and generative memory-processing costs with Qwen3.6-27B-FP8. Best values are bold and second-best values are underlined; 0 denotes no generative calls in that stage, and – indicates exclusion from resource comparisons. Final answer generation and evaluation are excluded from token costs.

Method	Performance (%) ↑		Build Tokens (M) ↓		Reasoning Tokens (M) ↓		Total (M) ↓	Calls ↓
	Judge	F1	In	Out	In	Out		
Dense RAG	46.93	24.86	–	–	–	–	–	–
LinearRAG	51.37	26.31	–	–	–	–	–	–
RAPTOR	40.10	23.25	4.412	1.321	**0**	**0**	5.733	5,333
MAGMA	50.16	23.91	3.028	0.878	**0**	**0**	3.906	5,094
LightMem	43.80	26.37	0.552	0.353	**0**	**0**	**0.905**	2,866
MemGAS	54.42	27.25	4.952	0.021	35.594	0.534	41.102	**564**
SimpleMem	46.29	22.90	3.697	0.846	0.397	0.189	5.128	1,620
LGMem (Ours)	**57.33**	**30.34**	**0**	**0**	4.993	0.507	5.500	771

LGMem achieves the highest Judge score (57.33) and token F1 (30.34) in Table 8. Compared with the second-best method, MemGAS, it improves Judge score by 2.92 percentage points and token F1 by 3.09 points while reducing generative memory-processing tokens by 86.6%. MemGAS uses fewer calls (564 versus 771), but its longer per-call contexts result in substantially higher token consumption. RAPTOR and SimpleMem have total token costs close to LGMem's 5.500M, yet their Judge scores are lower by 17.23 and 11.04 points, respectively; MAGMA uses 3.906M tokens but remains 7.17 points lower. LightMem records the lowest token cost because its user-only persistent-memory design processes only user turns, which account for 8.1% of the original BEAM-100K dialogue tokens before memory construction (Section E.3). This design reduces memory-processing cost but leaves information appearing only in assistant turns outside persistent memory, and its Judge score of 43.80 is 13.53 points below LGMem.

E.3 LightMem Input Coverage on BEAM

The 5.4–8.1% range reported in Section 3.2 denotes the fraction of original BEAM dialogue tokens contributed by user turns before memory construction: 8.1%, 7.1%, and 5.4% at 100K, 500K, and 1M, respectively. The declining fraction means that LightMem's user-only memory construction covers less of the source dialogue as history length grows. This narrower input is consistent with its low memory-processing cost, but excludes information that appears only in assistant turns from persistent memory.

E.4 Statistical Uncertainty

We use paired bootstrap resampling to quantify sampling variation in the primary Judge-score comparisons. LongMemEval-S resamples its 500 questions, while BEAM resamples complete dialogues so that questions from the same dialogue remain grouped. Each interval uses 100,000 resamples. Table 9 reports LGMem's difference from the strongest baseline in each main setting.

Table 9: Paired bootstrap 95% confidence intervals for the primary Judge-score differences. Differences are LGMem minus the strongest baseline and are reported in percentage points.

Setting	Baseline	Difference	95% CI
LongMemEval-S	LightMem	6.00	[1.60, 10.40]
BEAM-100K	MemGAS	6.01	[2.13, 10.06]
BEAM-500K	SimpleMem	8.52	[5.50, 11.58]
BEAM-1M	SimpleMem	9.70	[7.41, 11.91]

These intervals reflect variation over the evaluated questions or dialogues; they do not measure variation across repeated model or judge runs.

E.5 Sensitivity to Evidence-Selection Budgets

K controls the number of segments retrieved by each query. Because evidence accumulates across queries and reflection rounds, the pool may grow too large; K_f limits the final evidence selected for answer generation. We vary each budget separately in full BEAM-100K evaluations with GPT-5.6-Luna, holding the other budget and all remaining settings fixed.

Table 10: Sensitivity to the number of segments retrieved per query on BEAM-100K, with the final-evidence budget fixed at $K_f = 20$. Tokens and calls cover retrieval and reflection; Token F1 is diagnostic.

Retrieved segments K	**Judge (%)**	**Token F1 (%)**	**Query tokens (M)**	**Calls**	**Avg. rounds**
20	66.49	30.05	6.04	840	1.69
30 (selected)	67.12	30.26	6.40	891	1.68
40	65.78	29.90	7.09	979	1.73

Across $K = 20, 30, 40$, Judge scores stay within 1.34 percentage points, while query tokens increase from 6.04M to 7.09M. Neither alternative improves the observed Judge score over the default $K = 30$, so we retain the default.

Table 11: Sensitivity to the final-evidence budget on BEAM-100K, with the number of segments retrieved per query fixed at $K = 30$. Tokens and calls cover retrieval and reflection; Token F1 is diagnostic.

Final evidence K_f	**Judge (%)**	**Token F1 (%)**	**Query tokens (M)**	**Calls**	**Avg. rounds**
10	62.40	29.92	5.20	1211	1.82
20 (selected)	67.12	30.26	6.40	891	1.68
30	67.72	30.10	9.21	949	1.61

In this sweep, $K_f = 10$ yields a Judge score 4.72 points below the default. Increasing K_f from 20 to 30 produces a modest observed gain of 0.60 points while raising query tokens from 6.40M to 9.21M. We therefore retain the default $K = 30, K_f = 20$ as the more balanced operating point among the tested settings.

E.6 Runtime Analysis

Qwen3.6-27B-FP8 was served locally with vLLM in non-thinking mode on four NVIDIA L20 GPUs. We deployed two model replicas, each using tensor parallelism across two GPUs. We determined the highest stable concurrency for each method before the full evaluation; the reported runtimes therefore characterize complete-run throughput under method-specific stable concurrency rather than single-query latency. Runtime includes memory construction or indexing, retrieval and query reasoning, and final answer generation, whereas token cost includes only generative memory construction and query reasoning.

Table 12: Performance, token cost, and runtime on LongMemEval-S with Qwen3.6-27B-FP8. Best values are bold; – denotes an item excluded from the resource comparison.

Metric	**Dense RAG**	**Linear RAG**	**RAPTOR**	**MAGMA**	**MemGAS**	**LightMem**	**SimpleMem**	**LGMem**
Runtime (s) ↓	**794**	3,156	12,496	158,756	25,034	39,890	61,152	6,682
Judge Acc. (%) ↑	57.00	50.00	52.80	50.80	48.80	74.40	58.20	**77.20**
Token F1 (%) ↑	44.12	39.72	40.67	40.08	15.87	8.68	43.79	**60.08**
Token Cost (M) ↓	–	–	55.329	119.457	94.409	26.429	82.914	**6.218**

LGMem completes the full 500-question evaluation in 6,682 seconds. Dense RAG and LinearRAG are the only methods with shorter runtimes, but their Judge scores are 20.20 and 27.20 percentage

points lower, respectively. Compared with LightMem, which achieves the highest Judge accuracy among the baselines, LGMem gains 2.80 percentage points while reducing runtime and generative token cost by 83.2% and 76.5%, respectively. MAGMA requires 158,756 seconds (23.8× LGMem's runtime), whereas SimpleMem requires 61,152 seconds (9.2× LGMem's runtime). Their longer runtimes are consistent with heavier generative construction workloads: MAGMA uses 247,441 LLM calls to construct its message-level temporal graph, and SimpleMem processes 78.72M construction tokens for overlapping-window semantic compression, while LGMem builds its graph without LLM calls.

E.7 Cross-Session Retrieval Case

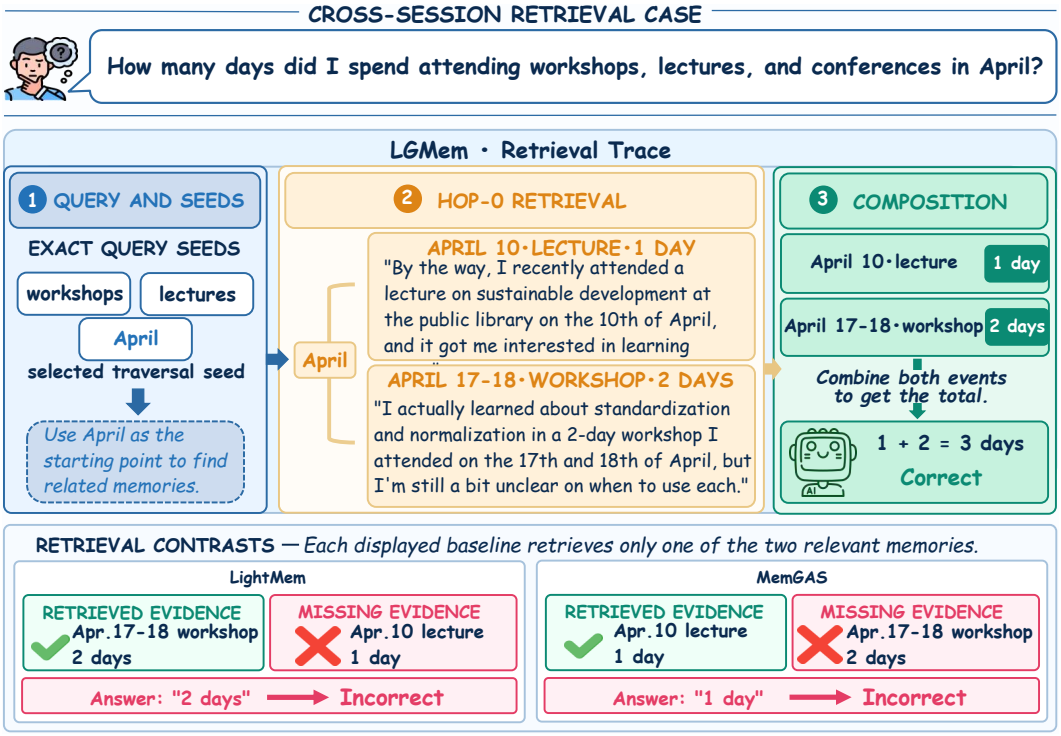


Figure 6: A case study of cross-session retrieval on LongMemEval-S. Starting from the exact *April* seed, LGMem retrieves a one-day lecture and a two-day workshop from different sessions at hop 0 and composes the three-day answer. LightMem and MemGAS each retrieve only one event, while SimpleMem (not shown) retrieves both.

Figure 6 illustrates a query whose answer requires combining events recorded in two different sessions. Starting from the exact *April* seed, LGMem retrieves both answer-bearing sentences at hop 0: a one-day lecture on April 10 and a two-day workshop on April 17–18. Their composition supports the three-day answer. The two displayed representative baselines recover complementary subsets of this evidence and therefore undercount the total, whereas SimpleMem, which is not shown, also retrieves both events and answers correctly.

F Prompts

Angle brackets denote runtime placeholders; bracketed IF/ELSE markers indicate conditional branches and are not included in the model input.

F.1 SHARED STATEFUL REFLECTION

Prompt 1: Stateful reflection

SYSTEM
You are a strict JSON assistant for retrieval planning. Return only one valid JSON object.

USER
We are improving retrieval for a memory QA system.
Your job is NOT to answer using outside knowledge. Inspect only the retrieved evidence.

Original question:
`<ORIGINAL QUESTION>`

[IF A QUESTION DATE IS AVAILABLE]
Question date / current date: `<NORMALIZED DATE>`
Use this date as question–time context when relevant.
[ELSE]
Question date / current date: unavailable.

Previous retrieval state:
`<PREVIOUS STATE OR empty>`

Previous retrieval queries:
`<JSON LIST OF PREVIOUS QUERIES>`

Current retrieved evidence:
`<NUMBERED EVIDENCE BLOCKS>`

Decide whether the current evidence is sufficient to answer the original question. Update the retrieval state: summarize known facts, propose a draft answer if possible, and identify exactly what information is still missing.

Be conservative when removing context: weak clues should be marked as possibly relevant, not irrelevant. Mark evidence as irrelevant only when it is clearly topic–only, generic advice, or not useful for answering the exact question.

If evidence is insufficient, generate retrieval subqueries that can find missing memory facts. Subqueries should be search phrases for evidence retrieval, not final answers. Preserve concrete constraints from the original question, such as entities, object types, actions, dates, people, locations, and preference words. Avoid generic subqueries that drop the topic or only ask for the answer type.

For count/list questions, split by counted actions or object types and include useful synonyms from the question. known_facts should list atomic candidate items/events/people and their evidence indices.

[IF THE QUESTION INVOLVES TIME, ORDER, COUNTS, LISTS, OR TOTALS]
For questions involving time, order, counts, lists, or totals, also populate structured_facts with atomic events/items/values. Use dates when available or inferable from evidence timestamps; leave date empty when no date is available.

OUTPUT FORMAT –– return JSON with exactly these keys:

```
{
  "can_answer": true or false,
  "confidence": number between 0 and 1,
  "draft_answer": "current best short answer, or empty string",
  "known_facts": [
    {"fact": "...", "evidence_indices": [1],
```

Prompt 1: Stateful reflection (continued)

```
      "answer_relevance": "direct|possible|background"}
  ],
  "structured_facts": [
    {"date": "YYYY-MM-DD or empty", "event_or_item": "...",
     "value": "...", "evidence_indices": [1],
     "relevance": "direct|possible|background"}
  ],
  "missing_information": ["..."],
  "subqueries": ["..."],
  "directly_relevant_evidence_indices": [1, 2],
  "possibly_relevant_evidence_indices": [3],
  "irrelevant_evidence_indices": [4],
  "irrelevant_evidence_confidence": {"4": 0.95},
  "rationale": "one short sentence"
}
```

At most <Kq> subqueries. Do not repeat previous retrieval queries.

F.2 Evidence Filtering

Prompt 2: Evidence filtering

SYSTEM
You are a strict JSON assistant for retrieval planning. Return only one valid JSON object.

USER
Filter retrieved memory evidence for final QA.
Be conservative: keep weak but potentially useful clues. Never remove evidence that may contain a count item, preference, date, place, or user−stated fact relevant to the question.

Original question:
<ORIGINAL QUESTION>

Protected evidence indices that should be kept unless clearly impossible:
<JSON LIST OF PROTECTED INDICES>

Evidence:
<NUMBERED EVIDENCE BLOCKS>

OUTPUT FORMAT −− return JSON:

```
{
  "keep_evidence_indices": [1, 2, 3],
  "remove_evidence_indices": [4],
  "rationale": "one short sentence"
}
```

Keep at most <FILTER TARGET K> evidence items unless more are clearly necessary.

F.3 Answer Generation

LongMemEval-S.

Prompt 3: LongMemEval-S unified answer prompt

SYSTEM
You are a diligent memory question–answering agent. Use only the retrieved conversation evidence. Gather and verify all relevant information silently, then output only the final short answer.

USER
You need to answer a question from retrieved memory evidence. Write a short answer with exact words from the event whenever possible.

[IF A QUESTION DATE IS AVAILABLE]
QUESTION CONTEXT
Question date / current date: `<NORMALIZED QUESTION DATE>`
Use this date as question–time context when relevant.

RETRIEVED EVIDENCE
`<NUMBERED RETRIEVED EVIDENCE; MAY INCLUDE A MARKED DERIVED-STATE BLOCK>`

QUESTION
`<ORIGINAL QUESTION>`

Rules:
– For yes/no or binary questions, output 'Yes', 'No', 'Likely yes', or 'Likely no'.
– For where/location/place questions, output a concrete and specific place name. If no exact name is mentioned, describe it briefly.
– For what/which questions, output one specific concrete item, event, subject, person, organization, place, or titled work directly asked for.
– For how many questions, output the number of tasks, objects, people, or events asked for; do not output the number of broad physical categories.
– For other questions, output only the minimal answer: a key phrase, entity, value, date, amount, or concise list.
– There may be multiple answers; synthesize all relevant retrieved evidence before answering.
– Write the answer with exact words or values from the evidence whenever possible.
– Do not include reasoning, explanations, citations, evidence IDs, or JSON.
– If the retrieved evidence is insufficient, answer exactly: Insufficient context.

[IF THE QUESTION IS TEMPORAL]
– For temporal questions, use absolute dates in the evidence and timestamp metadata to compute the requested time difference.
– For elapsed–time questions, compare the relevant event dates and answer in the unit requested by the question.
– For current/latest/most recent questions, choose the newest directly relevant event or value.

Final answer:

BEAM.

Prompt 4: BEAM unified answer prompt

SYSTEM
You are a memory question–answering assistant. Use only retrieved conversation evidence and timestamp metadata. Adapt the final answer form to the question, reason silently, and output only the answer.

USER
Answer this memory QA item using the retrieved conversation evidence. The context may contain raw turns, supporting evidence, and a retrieval–state summary.

[IF A QUESTION DATE IS AVAILABLE]
QUESTION CONTEXT
Question date / current date: `<NORMALIZED QUESTION DATE>`
Use this date as question–time context when relevant.

RETRIEVED EVIDENCE
`<NUMBERED RETRIEVED EVIDENCE; MAY INCLUDE A MARKED DERIVED-STATE BLOCK>`

QUESTION
`<ORIGINAL QUESTION>`

RULES
– Use only the retrieved evidence and timestamp metadata; do not use outside knowledge.
– Infer the required answer form from the question wording. Do not assume access to a category label.
– Return one final answer only. Do not include reasoning, citations, evidence IDs, or JSON.
– Be concise, but include all fields needed to make the answer complete.
– If direct evidence is insufficient, answer exactly: Insufficient context.
– For contradiction or conflict questions, explicitly state that the memories conflict, mention both conflicting statements, and say that confirmation is needed.
– For abstention or unavailable–information questions, do not infer; say Insufficient context when the requested fact is not directly present.
– For event–ordering questions, output the requested ordered events/items in order and preserve exact names and date clues when available.
– For information–extraction questions, output the exact value, name, date, item, or text span requested by the question.
– For knowledge–update or latest/current questions, compare old and updated values and answer with the latest directly supported value unless the question asks for the old value.
– For count, list, or multi–session questions, aggregate distinct matching facts across all evidence; include the count and a compact list of item names when that prevents ambiguity.
– For preference questions, answer with the supported preference or constraint profile, including likes, dislikes, avoid/heavy/lightweight constraints, and accepted options when relevant.
– For instruction–following questions, preserve the user–requested format, tool, style, or constraint exactly when evidence states it.
– For summarization questions, cover every requested dimension with a compact but complete answer; do not collapse multi–part summaries into a single vague phrase.
– For temporal questions, identify the relevant start and end events, use evidence dates or resolved timestamps, and answer in the requested unit.

[IF THE QUESTION IS TEMPORAL]
– Use absolute dates written in the evidence as event dates.
– Use conversation timestamps only to resolve relative dates or recency when the event text itself is relative.
– For elapsed–time questions, compute the elapsed time between the exact events named in the question and include endpoints if useful.

Prompt 4: BEAM unified answer prompt (continued)

Final answer: