BEYOND MEMORY CONSTRUCTION: RETHINKING MEMORY ACCESS FOR LLM-BASED CONVERSATIONAL AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

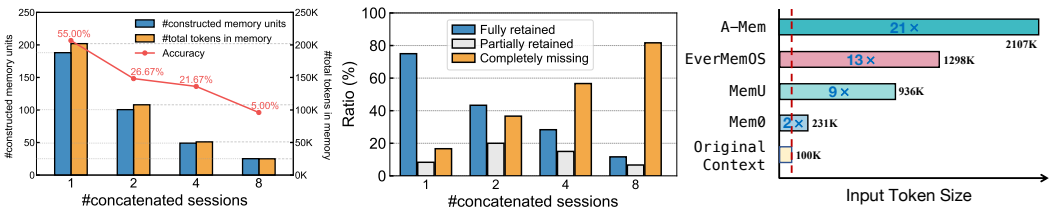
Memory is a core component of conversational agents, enabling coherent and context-aware behavior over long interactions. Recent approaches commonly rely on LLM-based memory construction, where raw interactions are rewritten into structured memory units and later retrieved via a RAG pipeline. While effective in controlled settings, we show that this paradigm breaks down in *long-horizon, high-entropy* conversations: memory construction becomes increasingly lossy and unstable as context length and information complexity grow, and incurs prohibitive cost due to repeated LLM invocation. To address these limitations, we propose `Threader`, a memory system that shifts the focus from memory construction to *efficient, structure-aware access over raw interactions*. Instead of rewriting interactions, `Threader` preserves them as first-class memory, organizes them into topic-coherent segments via lightweight incremental segmentation, and enables accurate retrieval through multi-view representation. At query time, it performs multi-signal retrieval that combines segment-level access with localized evidence matching, ensuring both completeness and coherence. Extensive experiments demonstrate that `Threader` consistently improves answer accuracy and evidence recall, while significantly reducing the memory construction overhead. Anonymous code is available at: https://anonymous.4open.science/r/Threader-C7B4/

1 INTRODUCTION

Large language model (LLM) agents have shown strong capabilities in multi-turn interactions, where maintaining and utilizing long-term memory is essential for coherent, personalized, and context-aware behavior. To support such capabilities, recent works (Zhang et al., 2025c; Hu et al., 2025b) commonly adopt an external memory paradigm: raw interactions are first transformed into structured memory units via LLM-based rewriting, summarization, or extraction, stored in a memory repository, and later retrieved through a retrieval-augmented generation (RAG) pipeline when answering new queries (Chhikara et al., 2025; Rasmussen et al., 2025; Xu et al., 2025; Hu et al., 2026). This design offers strong engineering advantages—it avoids modifying backbone models, generalizes across tasks, and can be readily deployed in practical systems.

However, we show that this paradigm encounters significant challenges when applied to **long-horizon, high-entropy** real-world conversational interactions, where dialogues are lengthy, topics are interleaved, and query-relevant evidence may appear sparsely and early in the conversation. First, **memory construction becomes lossy and unstable as conversation length and information entropy grow, leading to missing evidence and degraded QA performance.** While prior methods are typically evaluated on benchmarks such as LongMemEval (Wu et al., 2024), these settings significantly underestimate real-world interaction complexity: a single session in LongMemEval contains about 2.1K tokens, whereas real-world LLM usage has already exceeded an average of 5.4K tokens per request (Aubakirova et al., 2026). To bridge this gap, we simulate realistic scenarios by concatenating multiple sessions to construct longer and more entangled interaction histories. As shown in Figure 1a, our empirical results reveal that as the number of concatenated sessions increases, the representative Mem0 (Chhikara et al., 2025) method exhibits consistent degradation in downstream QA accuracy. This degradation is accompanied by a reduction in both the number of constructed memory units and their information content (quantified by the total number of tokens in

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.



(a) Accuracy and memory size of Mem0 (b) Evidence preservation in Mem0 (c) Memory construction cost

Figure 1: Limitations of existing LLM-based memory construction methods. As interaction length and information entanglement increase (simulated via session concatenation with LongMemEval), representative methods exhibit (a) degraded QA accuracy and shrinking memory size, (b) increasing loss of query-relevant evidence, and (c) substantial token amplification during memory construction.

their constructed memory units), indicating increasing information loss during memory construction. Further analysis in Figure 1b shows that a substantial portion of query-relevant evidence is either partially retained or completely missing. More details are provided in Appendix F.1. Second, **LLM-based memory construction incurs prohibitive cost**. These methods rely heavily on repeated LLM invocation during memory construction, leading to significant token amplification beyond the original input. As illustrated in Figure 1c, for a 100K-token interaction, the total LLM input tokens during construction can reach 2×, 9×, 13×, and 21× of the original input for Mem0 (Chhikara et al., 2025), MemU[1], EverMemOS (Hu et al., 2026), and A-Mem (Xu et al., 2025), respectively. This substantial overhead not only increases monetary cost but also limits scalability in long-horizon deployments.

These observations suggest that the limitations of existing methods stem from their reliance on *construction-time abstraction*, which is both lossy and expensive under long-horizon, high-entropy interactions. **This motivates a shift in design: instead of transferring interactions into memory during construction, we treat raw interactions as first-class memory with efficient query-time access.** This shift avoids construction-time information loss, reduces reliance on repeated LLM invocation, and enables more stable retrieval of query-relevant evidence. However, this introduces a key challenge: **how to precisely locate complete query-relevant evidence when it is entangled within long and mixed interaction histories?** In long-horizon, high-entropy settings, information is highly mixed and query-relevant evidence is often mixed with large amounts of irrelevant context, making accurate and complete retrieval difficult. Therefore, this requires organizing interactions in a way that preserves fine-grained evidence while supporting accurate and efficient access at query time.

This paper proposes `Threader`, a memory system designed for long-horizon, high-entropy conversational agents. `Threader` eliminates LLM-based abstraction during memory construction. Instead, it organizes interaction logs into *lossless, topic-coherent segments* through a lightweight and incremental segmentation process, preserving fine-grained evidence. It further constructs *multi-view segment representations* that capture semantic, temporal, and structural cues, enabling precise retrieval. At query time, the system performs *multi-signal evidence retrieval* to accurately identify relevant segments by combining segment-level access with localized evidence matching, providing coherent and evidence-complete context for response generation.

We evaluate `Threader` on multiple long-context conversational benchmarks. Experimental results show that `Threader` consistently outperforms advanced agent memory systems and agentic RAG methods in answer accuracy and evidence recall. Moreover, it significantly reduces both memory construction overhead and query-time latency, achieving efficient memory access.

Our contributions are summarized as follows: (1) We identify fundamental limitations of LLM-based memory construction in long-horizon, high-entropy conversational settings, supported by empirical analysis on both effectiveness and cost. (2) We propose `Threader`, a memory framework that shifts the focus from memory construction to efficient, structure-aware access over raw interactions. (3) We design a unified system combining incremental segmentation, multi-view representation, and multi-signal retrieval for robust evidence access. (4) We demonstrate that `Threader` achieves superior performance while substantially reducing both construction and query-time cost.

[1]MemU is an open-source memory system: https://github.com/NevaMind-AI/memU

2 Related Work

Agent Memory Paradigms. Recent taxonomies (Hu et al., 2025b) categorize agent memory into three primary architectures: non-parametric (token-level), parametric, and latent memory. Non-parametric memory stores explicit units, such as chunks or summaries, in external datastores. Systems like MemoryBank (Zhong et al., 2024), MemGPT (Packer et al., 2023) and Mem0 (Chhikara et al., 2025) represent this dominant paradigm, offering interpretability and ease of updates. In contrast, parametric memory stores information in model weights, often through editing (Wang et al., 2021; Meng et al., 2022; Wang et al., 2024a), whereas latent memory relies on internal representations or recurrent states (Wang et al., 2024b; 2025; Zhang et al., 2025a). However, parametric and latent methods often face challenges in update cost, controllability, and interpretability, making them less suitable for rapidly evolving dialogue memory. Thus, token-level memory remains a natural choice for long-context human–assistant dialogue, where memories must be explicit, updatable, and retrievable.

Factual Memory in Dialogue. Within the non-parametric line, prior work often distinguishes experiential memory, which stores reusable strategies, skills, or trajectories for future task solving (Shinn et al., 2023; Zhao et al., 2024; Wang et al., 2024c), from working memory, which maintains transient context for ongoing reasoning within a task (Yu et al., 2025; Wu et al., 2025; Hu et al., 2025a). For sustained human-assistant interaction, however, factual memory—storing user-specific facts and interaction history—remains the central challenge (Rasmussen et al., 2025; Kang et al., 2025; Li et al., 2025; Hu et al., 2026). Most factual-memory systems follow a common pipeline: they rewrite raw dialogue into compact memory units and retrieve them at query time. This reliance on LLM-mediated memory rewriting introduces two practical limitations: memory construction can become increasingly lossy and unstable in realistic long-context dialogue, and the repeated LLM calls required for rewriting impose substantial construction cost.

RAG for Memory. Agent memory differs from classical RAG mainly in persistence: memory stores and reuses information across interactions, rather than only retrieving external knowledge for inference (Gao et al., 2023; Singh et al., 2025). HippoRAG (Gutiérrez et al., 2024) and HippoRAG2 (Gutiérrez et al., 2025) connect retrieval with long-term memory, showing that memory can be viewed as structured non-parametric retrieval. Similarly, many factual-memory systems for long-context dialogue store histories in external banks and retrieve relevant units at query time (Packer et al., 2023; Chhikara et al., 2025; Rasmussen et al., 2025; Hu et al., 2026). Unlike conventional RAG, conversational memory must first organize evolving interactions into retrievable units. We therefore treat memory formation not as semantic rewriting, but as evidence-preserving structural organization for efficient access.

3 Methodology

As illustrated in Fig. 2, `Threader` makes raw interaction histories usable as long-term memory through structure-aware access rather than LLM-based rewriting. This requires solving three challenges: identifying coherent evidence boundaries, representing heterogeneous evidence, and retrieving complete evidence from long and entangled histories. Accordingly, `Threader` consists of three components. **(1) Incremental Topic-Coherent Segmentation** partitions interaction histories into semantically consistent units while preserving the original fine-grained evidence. **(2) Multi-View Segment Representation** encodes each segment from complementary perspectives to avoid collapsing user-side, assistant-side, and full-context evidence into a single representation. **(3) Multi-Signal Evidence Retrieval** combines dense, lexical, and reranking signals to locate query-relevant evidence and reconstructs the retrieved segments into temporally consistent context for response generation.

3.1 Incremental Topic-Coherent Segmentation

In long-horizon conversational settings, interaction histories are often *high-entropy*, where multiple topics are interleaved and query-relevant evidence is sparsely distributed. A common practice in RAG systems is to partition text into fixed-size chunks. However, such chunking is agnostic to semantic boundaries, often fragmenting coherent evidence or mixing unrelated content, which hinders precise retrieval. Instead, we aim to segment interactions into *topic-coherent evidence segments*, such that each segment forms a semantically consistent unit that can be retrieved as a whole.

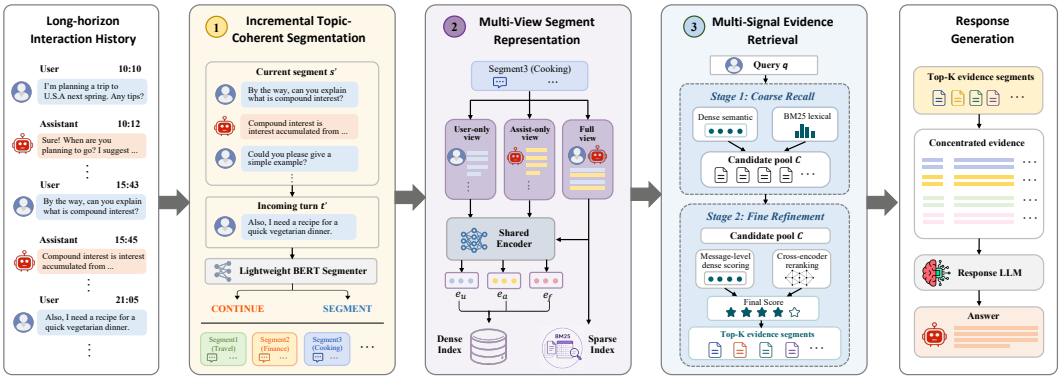


Figure 2: **Overview of `Threader`.** It incrementally segments interaction streams into topic-coherent units, encodes them with multi-view representations, and retrieves related evidence with multi-signals.

Let a conversation be denoted as $C = \{S_1, \ldots, S_n\}$, where each session S_i is an ordered sequence of turns: $S_i = (t_1, t_2, \ldots, t_m)$. Our goal is to partition each session into a sequence of contiguous, topic-coherent segments: $\mathcal{S}_i = (s_1, s_2, \ldots, s_K)$, $s_k = (t_{k,1}, t_{k,2}, \ldots, t_{k,n_k})$, where each segment s_k is a contiguous subsequence of turns sharing a consistent semantic topic.

An effective segmenter for conversational memory should satisfy two key properties: 1) **Lightweight**, to avoid the high cost of LLM-based preprocessing; 2) **Incremental**, to support streaming interactions without requiring full-context recomputation. Prior work has explored topic-aware segmentation using LLMs (e.g., SECOM (Pan et al., 2025), EverMemOS (Hu et al., 2026)), but such approaches incur substantial computational overhead. More lightweight methods (e.g., LightMem (Fang et al., 2025)) rely on embedding or attention-based heuristics, but are typically not fully incremental, as they require buffering and batch processing before determining segment boundaries.

To achieve both efficiency and incrementality, we formulate segmentation as a *binary decision problem* over streaming turns. The system maintains a current segment s' (temporal history), and processes incoming turns sequentially. Given a new turn t', the model predicts whether t' should be appended to s' or start a new segment: CONTINUE: append t' to the current segment; SEGMENT: finalize the current segment and start a new one with t'. Formally, let $s' = (t'_1, \ldots, t'_j)$ denote the current segment. For an incoming turn t'_{j+1}, we construct an input: $x = \text{Concat}(s', [\text{SEP}], t'_{j+1})$, and predict a binary label $y \in \{0, 1\}$ indicating CONTINUE or SEGMENT. This formulation enables fully online segmentation without revisiting past context.

We instantiate the segmenter using a lightweight BERT encoder (Devlin et al., 2019), which provides strong bidirectional semantic modeling while maintaining low computational overhead. We construct training labels using both coarse-grained and fine-grained signals. Session boundaries from LongMemEval (Wu et al., 2024) are used as *hard labels*, capturing clear conversational shifts. To further refine intra-session topic boundaries, we follow prior work and leverage LLM-based annotation to obtain *soft labels* (Pan et al., 2025). This combination allows the model to capture both session-level transitions and finer-grained topic changes.

Given segmented conversations, we construct binary classification examples for incremental decision making. For turns within the same segment, we create CONTINUE examples:

$$\left(x^{\text{cont}}_{k,j},\, y^{\text{cont}}_{k,j}\right) = \left(\text{Concat}((t_{k,1}, \ldots, t_{k,j}), [\text{SEP}], t_{k,j+1}),\, 0\right), \quad j = 1, \ldots, n_{k-1} \tag{1}$$

For segment boundaries, we create SEGMENT examples:

$$\left(x^{\text{seg}}_{k},\, y^{\text{seg}}_{k}\right) = \left(\text{Concat}((t_{k,1}, \ldots, t_{k,n_k}), [\text{SEP}], t_{k+1,1}),\, 1\right), \quad k = 1, \ldots, K-1 \tag{2}$$

The model is trained using the standard cross-entropy loss. Full details are provided in Appendix F.2.

3.2 Multi-View Segment Representation

A fundamental challenge in long-horizon conversational retrieval is that evidence is *heterogeneous*: user utterances often encode persistent and query-relevant facts (e.g., preferences, experiences), while assistant responses are typically longer, more verbose, and contain substantial generative content. A

naive approach is to encode each segment into a single embedding. However, this inevitably collapses heterogeneous evidence into a unified representation, where dominant assistant-side content can overshadow weak but critical user-side signals, degrading retrieval of query-relevant evidence.

To address this, we adopt a *multi-view representation* that explicitly separates different evidence sources. Given a segment $s_i = (t_{i,1}, \ldots, t_{i,m})$ with turns $t_{i,j} = (u_{i,j}, a_{i,j})$, we construct three complementary views: $x_i^{(u)} = \mathrm{Concat}(u_{i,1}, \ldots, u_{i,m}), x_i^{(a)} = \mathrm{Concat}(a_{i,1}, \ldots, a_{i,m}), x_i^{(f)} = \mathrm{Concat}(u_{i,1}, a_{i,1}, \ldots, u_{i,m}, a_{i,m})$, corresponding to user-only, assistant-only, and full interaction representations. Each view is encoded using a shared encoder f_{enc}: $e_i^{(v)} = f_{\mathrm{enc}}(x_i^{(v)}), v \in \{u, a, f\}$. This design decouples evidence into complementary representations, allowing retrieval to selectively attend to user-centric signals while preserving contextual completeness.

3.3 Multi-Signal Evidence Retrieval

Given a query q and segments $\{s_i\}$, we retrieve segments that jointly maximize evidence relevance and completeness. We formulate this as a *multi-signal relevance estimation problem*, where different signals capture complementary aspects of evidence. Specifically, we consider three retrieval signals.

Given the query embedding $z(q) = f_{\mathrm{enc}}(q)$, the **dense semantic signal** aggregates similarities across the three segment views:

$$\mathrm{score}_i^{(e)}(q) = \sum_{v \in \{u,a,f\}} w_v \cos\big(z(q), e_i^{(v)}\big), \quad \sum_v w_v = 1. \tag{3}$$

We complement this with a **lexical matching signal**, $\mathrm{score}_i^{(b)}(q) = \mathrm{BM25}(q, s_i)$, and a **cross-encoder signal**, $\mathrm{score}_i^{(r)}(q) = \mathrm{Rerank}(q, s_i)$. Dense and lexical scores support efficient candidate retrieval, while the cross-encoder captures fine-grained query–segment interactions at higher computational cost, motivating a *coarse-to-fine two-stage retrieval strategy*.

Stage I: High-recall segment retrieval. We first construct a high-recall candidate pool $\mathcal{C}$, which provides a shortlist of segments for fine-grained relevance estimation in Stage II. Specifically, we take the union of the top-K_1 segments retrieved by dense retrieval and BM25:

$$\mathcal{C} = \text{Top-}K_1(\mathrm{score}^{(e)}(q)) \cup \text{Top-}K_1(\mathrm{score}^{(b)}(q)). \tag{4}$$

This combines semantic coverage with lexical matching to improve recall.

Stage II: Evidence-level refinement. A key limitation of segment-level retrieval is that query-relevant evidence is often *locally concentrated within segments*, rather than uniformly distributed across all turns. To capture such fine-grained signals, we refine dense matching from segment-level to message-level scoring.

Concretely, let candidate segment s_i contain m_i turns, denoted as $\{t_{i,j} = (u_{i,j}, a_{i,j})\}_{j=1}^{m_i}$. For each turn, we compute the similarity between the query and each user/assistant message using the encoder f_{enc}; message embeddings are pre-computed during indexing. This produces a set of message-level relevance scores within the segment. We then select the Top-K_2 highest scores and average them to obtain a refined dense score:

$$\widetilde{\mathrm{score}}_i^{(e)}(q) = \frac{1}{K_2} \sum_{k=1}^{K_2} \mathrm{TopK}_k\big(\{\cos(z(q), f_{\mathrm{enc}}(u_{i,j})), \cos(z(q), f_{\mathrm{enc}}(a_{i,j}))\}_{j=1}^{m_i}\big). \tag{5}$$

Before each fusion step, scores are min-max normalized over candidate segments for comparable scales. We then combine the normalized dense signal with lexical matching:

$$\mathrm{score}_i^{(h)} = \widetilde{\mathrm{score}}_i^{(e)}(q) + \beta\, \mathrm{score}_i^{(b)}(q), \tag{6}$$

and further apply cross-encoder reranking to obtain the final score:

$$\mathrm{score}_i = \alpha\, \mathrm{score}_i^{(h)} + (1 - \alpha)\, \mathrm{score}_i^{(r)}(q). \tag{7}$$

From the final score score_i, we select the top-K segments for downstream response generation. In particular, although Stage II operates at message-level granularity, we ultimately perform retrieval at the segment level. This design preserves *discourse coherence* in downstream reasoning, as query-relevant evidence in conversational agents is often meaningful only within its surrounding context. In this way, fine-grained scoring is used to identify evidence within segments, while segment-level retrieval ensures that retrieved content remains structurally and temporally consistent for generation.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Datasets and Evaluation. We evaluate on **LongMemEval** (Wu et al., 2024) and **PersonaMem** (Jiang et al., 2025), two benchmarks that capture complementary long-term memory settings. LongMemEval contains 500 QA instances with an average session length of 2K tokens, covering information extraction, multi-session reasoning, temporal reasoning, and knowledge updates. PersonaMem provides a more challenging personalization setting with substantially longer sessions (589 QA instances, 26K tokens on average), emphasizing long-horizon user-specific memory. Dataset details are provided in Appendix D. We evaluate LongMemEval using **LLM-as-a-Judge** and PersonaMem using exact match. Detailed judge protocols are provided in Appendix F.3.

Baselines. We compare `Threader` with seven representative baselines, including five *agent memory* methods and two *agentic RAG* approaches. The memory baselines include **Mem0** (Chhikara et al., 2025), **MemU**, **A-Mem** (Xu et al., 2025), **MemOS** (Li et al., 2025), and **EverMemOS** (Hu et al., 2026), covering recent advances in memory construction, organization, updating, and retrieval. The agentic RAG baselines include **Self-RAG** (Asai et al., 2024) and **HippoRAG2** (Gutiérrez et al., 2025), which represent strong retrieval-based agents without explicit memory construction. Details of the baselines are available in Appendix E.

Implementation Details. Following our lightweight design principle, `Threader` uses paraphrase-multilingual-MiniLM-L12-v2 (Reimers & Gurevych, 2019) for embedding and bge-reranker-base (Xiao et al., 2023) for reranking. Our main setting uses GPT-4.1-mini for memory construction and GPT-5-mini for response generation, reflecting practical deployments that prioritize cost-efficient construction and high-quality reasoning. We also report a low-latency setting with GPT-4.1-mini and GPT-5-nano. Results with Qwen models are provided in Appendix F.4. For fair comparison, all LLM-based construction methods share the same construction model, and all methods share the same response model while retaining method-specific components.

4.2 MAIN RESULTS

Table 1 and Table 2 report the main results. Overall, `Threader` consistently outperforms all baselines across both datasets and response backbones, demonstrating the effectiveness of structure-aware access over LLM-based memory construction or retrieval in long-horizon conversational settings. On LongMemEval, `Threader` improves over the strongest baseline by 3.33% with GPT-5-nano and 6.84% with GPT-5-mini. On PersonaMem, `Threader` also obtains the best overall score, with relative gains of 4.50% and 7.29%, respectively.

On LongMemEval, the performance gains are especially pronounced on evidence-intensive question types. `Threader` improves *Temporal-Reasoning* by 6.94% with GPT-5-nano and 11.02% with GPT-5-mini, and improves *Knowledge-Update* by 5.80% with GPT-5-mini. It also achieves competitive results on *Multi-Session*, *Single-Session-User*, and *Single-Session-Assistant* tasks. These tasks require locating sparse and temporally distant evidence, where lossy memory construction often fails. In contrast, `Threader` preserves fine-grained interaction structure and retrieves topic-coherent segments, leading to more complete evidence coverage. We observe a similar trend on PersonaMem, where `Threader` achieves the largest improvements on *generalize-to-new-scenarios* and *recalling-user-shared-facts*, further validating its advantage in long and entangled conversational histories.

`Threader` is less advantageous on tasks such as Single-Session-Preference (LongMemEval) and *tracking-full-preference-evolution* (PersonaMem), which require aggregating and abstracting user preferences across interactions. Unlike memory-construction approaches that explicitly summarize preferences, `Threader` operates on raw evidence and relies on the response model for higher-level synthesis. This suggests a complementary trade-off between evidence preservation and abstraction.

Overall, the results indicate that `Threader` is particularly effective when accurate memory access depends on preserving and retrieving distributed evidence, which is the dominant challenge in long-horizon, high-entropy conversational settings.

Table 1: **Main results on LongMemEval.** `Threader` retrieves the top-12 segments for LongMemEval. SS-User, SS-Assi., SS-Pref., Multi-S., Know.-U., and Temp.-R. denote Single-Session-User, Single-Session-Assistant, Single-Session-Preference, Multi-Session, Knowledge-Update, and Temporal-Reasoning, respectively. Results are grouped by backbone. Within each backbone block, the best result in each column is shown in **bold**, the second-best distinct result is underlined.

Method	SS-User	SS-Assi.	SS-Pref.	Multi-S.	Know.-U.	Temp-R.	Overall
			GPT-5-nano				
Self-RAG	87.14	94.64	43.33	56.39	73.08	73.68	71.40
HippoRAG2	71.43	94.64	53.33	60.15	76.92	69.92	70.40
Mem0	80.00	46.43	26.67	39.85	69.23	58.65	55.00
MemU	81.43	78.57	43.33	55.64	53.85	68.42	64.20
A-Mem	84.29	82.14	20.00	30.83	56.41	27.82	46.60
MemOS	81.43	87.50	**100.00**	57.89	71.79	45.11	65.80
EverMemOS	92.85	78.57	90.00	66.92	82.05	75.93	78.00
Threader	**94.29** (↑1.55%)	**98.21** (↑3.77%)	63.33 (↓36.67%)	**67.67** (↑1.12%)	**83.33** (↑1.56%)	**81.20** (↑6.94%)	**80.60** (↑3.33%)
			GPT-5-mini				
Self-RAG	92.86	96.43	60.00	61.65	78.21	77.44	76.60
HippoRAG2	72.86	92.86	70.00	67.67	87.18	69.92	75.00
Mem0	81.43	42.86	26.67	44.36	71.79	61.65	57.20
MemU	87.14	82.14	40.00	56.39	79.49	76.69	71.60
A-Mem	88.57	**98.21**	43.33	51.13	76.92	58.65	67.20
MemOS	82.86	89.29	**100.00**	60.90	79.49	45.86	68.40
EverMemOS	95.71	80.35	96.67	78.95	88.46	81.95	84.80
Threader	**98.57** (↑2.99%)	**98.21** (→0.00%)	96.67 (↓3.33%)	**79.70** (↑0.95%)	**93.59** (↑5.80%)	**90.98** (↑11.02%)	**90.60** (↑6.84%)

Table 2: **Main results on PersonaMem.** `Threader` retrieves the top-8 segments for PersonaMem. Generalize-New-Scn., Preference-Align, User-Shared, User-Mentioned, Update-Reason, New-Ideas, and Preference-Evolution denote generalizing-to-new-scenarios, provide-preference-aligned-recommendations, recalling-user-shared-facts, recalling-facts-mentioned-by-the-user, recalling-reasons-behind-previous-updates, suggesting-new-ideas, and tracking-full-preference-evolution, respectively. Results are grouped by backbone. Within each backbone block, the best result in each column is shown in **bold**, the second-best distinct result is underlined.

Method	Generalize -New-Scn.	Preference -Align	User-Shared	User-Mentioned	Update-Reason	New-Ideas	Preference Evolution	Overall
				GPT-5-nano				
Self-RAG	29.82	52.73	16.28	47.06	67.68	38.71	48.20	41.60
HippoRAG2	56.14	65.45	53.49	**70.59**	69.70	**39.78**	51.08	55.35
Mem0	45.61	43.64	31.01	47.06	70.71	**39.78**	55.40	47.88
MemU	35.09	41.82	47.29	41.18	58.59	27.96	51.80	45.33
A-Mem	49.12	67.27	60.47	52.94	63.64	29.03	30.94	48.39
MemOS	63.16	69.09	68.22	58.82	76.77	34.41	54.68	60.44
EverMemOS	50.88	61.82	58.91	64.71	**77.78**	38.71	51.80	56.88
Threader	**78.95** (↑25.00%)	**72.73** (↑5.27%)	**69.77** (↑2.27%)	52.94 (↓25.00%)	72.73 (↓6.49%)	37.63 (↓5.40%)	**58.27** (↑5.18%)	**63.16** (↑4.50%)
				GPT-5-mini				
Self-RAG	29.82	52.73	4.65	58.82	55.56	**48.39**	54.68	40.41
HippoRAG2	61.40	63.64	68.22	**70.59**	67.68	37.63	56.12	59.42
Mem0	57.89	41.82	29.46	35.29	68.69	27.96	**66.91**	48.73
MemU	54.39	41.82	60.47	52.94	67.68	27.96	61.87	54.33
A-Mem	70.18	65.45	70.54	64.71	72.73	29.03	58.27	60.78
MemOS	71.93	**72.73**	72.87	58.82	82.83	30.11	61.87	64.69
EverMemOS	68.42	67.27	74.42	**70.59**	**86.87**	26.88	64.03	65.20
Threader	**84.21** (↑17.07%)	70.91 (↓2.50%)	**80.62** (↑8.33%)	**70.59** (→0.00%)	79.80 (↓8.14%)	45.16 (↓6.67%)	63.31 (↓5.38%)	**69.95** (↑7.29%)

4.3 Ablation Studies

As shown in Table 3, we conduct a comprehensive ablation study to evaluate the key design axes of `Threader`, including (i) topic-coherent segmentation, (ii) multi-view segment representation, (iii) multi-signal retrieval, and (iv) retrieval granularity (segment-level in `Threader` vs. message-level).

Table 3: **Ablation study on the key design axes of `Threader`.** We evaluate the impact of segmentation, multi-view representation, multi-signal retrieval, and retrieval granularity on answer accuracy and evidence recall under Top-5 and Top-12 settings. Full-R denotes the fraction of queries with complete evidence coverage, and Mean-R denotes average evidence recall. Emb.-User, Emb.-Asst., and Emb.-Full correspond to using only user, assistant, or full interaction views, respectively.

Variant	Top-5			Top-12		
	ACC	Full-R	Mean-R	ACC	Full-R	Mean-R
Segmentation ablation						
`Threader`	82.60	82.55	90.26	90.60	93.40	96.50
w/o Seg.	82.00	–	–	87.00	–	–
Chunk-500	82.40	–	–	85.60	–	–
Chunk-1K	74.20	–	–	86.40	–	–
Chunk-2K	63.60	–	–	79.80	–	–
Multi-view representation ablation						
Emb. w/o Msg.	71.20	67.45	77.34	85.40	86.81	91.65
Emb. - Full	70.20	64.68	75.52	84.00	82.55	88.72
Emb. - User	68.20	61.91	73.45	83.00	82.76	89.07
Emb. - Asst.	62.40	55.11	68.59	75.40	73.40	82.88

Variant	Top-5			Top-12		
	ACC	Full-R	Mean-R	ACC	Full-R	Mean-R
Retrieval signal ablation						
`Threader`	82.60	82.55	90.26	90.60	93.40	96.50
w/o Emb.	79.00	79.15	87.40	84.60	86.17	92.19
w/o BM25	81.60	80.43	88.21	88.20	91.49	94.81
w/o Rerank.	78.40	77.23	86.43	88.20	91.49	95.08
w/o Emb.&BM25	80.40	76.17	85.31	85.60	88.51	93.54
w/o Emb.&Rerank.	74.80	73.19	81.83	83.20	84.68	91.06
w/o BM25&Rerank.	72.00	69.57	80.63	87.00	89.15	93.76
w/o Msg.-Emb.	81.80	81.28	88.72	89.20	93.19	96.29
Retrieval granularity ablation						
Segment Unit	82.60	82.55	90.26	90.60	93.40	96.50
Message Unit	82.00	–	–	85.80	–	–

These ablations are designed to test how different components contribute to robust information access under entangled conversational settings.

Segmentation ablation. Disabling segmentation or replacing topic-coherent segmentation with fixed-size chunking consistently degrades performance, suggesting that retrieval quality depends on semantically aligned evidence units rather than coverage alone. Coherent segments provide more faithful boundaries for evidence, reducing fragmentation and improving downstream matching. **Multi-view representation ablation.** We isolate multi-view representation under dense-only retrieval by disabling message-level reranking. Removing views from the multi-view encoding leads to noticeable performance drops. This indicates that collapsing heterogeneous interaction signals into a single representation harms retrieval, as weaker but important user-side evidence can be overshadowed. Multi-view representations mitigate this effect by separating complementary information sources. **Multi-signal retrieval ablation.** We observe that eliminating individual retrieval signals degrades performance, while removing their combination leads to larger drops. This confirms that different signals capture complementary aspects of relevance—semantic similarity, lexical matching, and fine-grained interaction—and their integration is critical for robust retrieval. **Retrieval granularity ablation.** To evaluate the effectiveness of the segment-level retrieval strategy adopted in `Threader`, we compare with message-level retrieval (i.e., treating each user or assistant message as an independent retrieval unit instead of full segments). The results show that, while fine-grained message-level matching improves the ability to locate relevant signals, retrieving individual messages alone degrades performance. This indicates that although relevant information is often localized, it typically relies on surrounding context for correct interpretation. Segment-level retrieval provides a better trade-off by preserving contextual coherence while enabling fine-grained evidence localization through message-level scoring. **Overall**, these results demonstrate that the effectiveness of `Threader` arises from the joint design of structure-aware segmentation, multi-view representation, and multi-signal retrieval, which together enable accurate and stable access to relevant information.

4.4 Hyperparameter Analysis

We analyze the effect of the main retrieval hyperparameter, the number of retrieved segments K, in Figure 3. As K increases, `Threader` can access more potentially relevant evidence, leading to consistent accuracy improvements in the early stage. However, retrieving more segments is not always beneficial: after K becomes sufficiently large, additional segments introduce more irrelevant or weakly related context, and the accuracy no longer increases steadily but starts to fluctuate. Notably, `Threader` already surpasses all state-of-the-art baselines with a relatively small retrieval budget. On LongMemEval, `Threader` outperforms all baselines at Top-7,

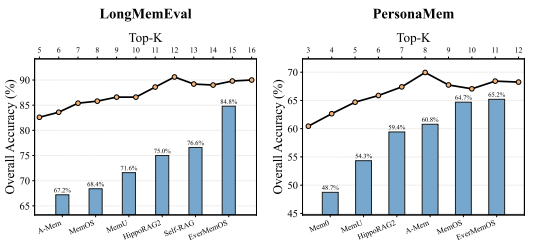


Figure 3: **Top-K sensitivity analysis on LongMemEval and PersonaMem.**

Table 4: **Cost comparison of agent memory systems across offline memory construction and online query-time serving.** MemOS is omitted due to its black-box interface, which prevents reliable cost attribution. We report construction cost per 100K input tokens and query-time cost per query, covering runtime/latency (s), LLM calls, input and output token consumption, and monetary cost ($). The best and worst reported results are highlighted in green and red, respectively.

Method	Memory Construction					Query-Time				
	Time	#Calls	In Tok.	Out Tok.	Cost $	Latency	#Calls	In Tok.	Out Tok.	Cost $
Self-RAG	13.18	0	0	0	0	93.52	6.55	2.3K	0.4K	0.0016
HippoRAG2	110.84	95.26	249K	38K	0.1604	51.02	3.03	17.5K	0.7K	0.0081
Mem0	1239.62	47.94	231K	64K	0.1948	8.38	0	0	0	0
MemU	2448.94	305.65	936K	121K	0.5680	199.47	19.09	16.1K	2.5K	0.0112
A-Mem	2032.45	1415.19	2107K	79K	0.9692	12.35	1.00	9.9K	0.9K	0.0054
EverMemOS	2013.97	632.42	1298K	146K	0.7528	53.68	1.75	3.1K	0.3K	0.0017
Threader	18.33	0	0	0	0	8.66	0	0	0	0

and on PersonaMem, it surpasses all baselines at Top-6. This suggests that Threader is not only effective at high retrieval budgets, but can also locate useful evidence with a compact retrieved context. We further analyze the retrieval hyperparameters in Eqs. 3, 6, and 7 in Appendix F.6.

4.5 Cost Analysis

Cost is a critical yet under-reported factor in agent memory systems. Prior work often reports a single aggregated number, obscuring the fundamentally different cost characteristics of memory construction and query-time serving. We therefore decompose system cost into two stages: **(1) offline memory construction**, where monetary cost (LLM calls, token usage) dominates, and **(2) online response**, where latency is the primary concern.

Table 4 shows that existing methods incur substantial cost in at least one stage. Rewrite-heavy approaches introduce significant construction overhead. For example, processing 100K input tokens requires **21.1**× and **13.0**× token expansion for A-Mem and EverMemOS, respectively, along with hundreds of LLM calls and non-trivial monetary cost. In contrast, Threader eliminates construction-time LLM usage and builds memory indices in 18.33s, achieving over **110**× speedup compared to these methods. It is only slightly slower than Self-RAG (18.33s vs. 13.18s), mainly due to its additional topic-coherent segmentation step before indexing. At query time, Threader maintains low latency (8.66s) without retrieval-side LLM calls, incurring no extra token cost beyond final response generation. Although Mem0 is slightly faster (8.38s) due to simpler retrieval over pre-written memory entries, Threader achieves comparable latency without LLM-based construction. Despite its multi-signal design (Section 3.3), Threader's retrieval remains lightweight in practice, achieving **5.9**×**–23.0**× query-time speedups over other representative baselines. Overall, existing systems face a trade-off: LLM-based memory construction incurs high offline cost, while agentic retrieval introduces expensive query-time overhead. Threader avoids this trade-off by replacing LLM-based construction with lightweight structure-aware indexing and using LLM-free retrieval, enabling both low construction overhead and efficient query-time access.

5 Conclusions

In this work, we revisit the design of memory systems for conversational agents and identify fundamental limitations of LLM-based memory construction in long-horizon, high-entropy settings. Through empirical analysis, we show that heavy reliance on construction-time abstraction leads to both information loss and substantial computational overhead, limiting the effectiveness and scalability of existing approaches. We propose Threader, a memory framework that shifts the focus from constructing compressed representations to enabling efficient, structure-aware access over raw interactions. By combining incremental segmentation, multi-view representation, and multi-signal retrieval, Threader achieves robust and stable access to query-relevant evidence while preserving contextual coherence. Extensive experiments demonstrate that Threader not only improves answer accuracy and evidence recall, but also substantially reduces both construction and query-time cost.

AI USE STATEMENT

The authors designed the algorithmic framework of THREADER and implemented the method. Generative AI tools were used to assist in reviewing the framework and implementation, identifying potential errors, and suggesting refinements. The authors assessed these suggestions, determined which changes to incorporate, and reviewed and tested the resulting implementation.

The manuscript was drafted by the authors. Generative AI tools were used to assist with wording, organization, and technical presentation, and to improve grammar, clarity, and readability. The authors reviewed and revised the AI-assisted edits to ensure consistency with their intended meaning, the implemented method, and the reported experimental results.

As part of the experimental pipeline, GPT-4.1-mini was used to annotate fine-grained intra-session topic boundaries for segmenter training and evaluation. GPT-4.1-mini was also used as an LLM-as-a-Judge to evaluate answer correctness on LongMemEval by comparing generated responses against reference answers under the specified grading protocol. These uses are described in the corresponding annotation and evaluation sections.

The research premises and hypotheses were established by the authors. The authors made all final scientific decisions regarding the method, experimental design, interpretation of results, and conclusions. The authors take full responsibility for the final content of this work, including its methods, analyses, results, claims, and AI-assisted materials.

ETHICS STATEMENT

We strictly follow the data-usage and security requirements when interacting with the OpenAI API and Alibaba Qwen API services. This study uses only publicly available datasets and does not involve collecting new personal data from participants. To mitigate potential information leakage, we restrict API inputs to benchmark content, remove identifying information where applicable, and enable provider-side privacy controls (e.g., disabling data sharing or retention for model improvement when available). All API credentials are securely managed, and model outputs are used solely for academic research and evaluation purposes.

REPRODUCIBILITY STATEMENT

Anonymous code is available at https://anonymous.4open.science/r/Threader-C7B4/, as linked in the abstract. The paper describes the algorithmic framework in Section 3 and provides pseudocode in Appendix C. Implementation details, segmenter training procedures, and retrieval hyperparameters are documented in Appendices E, F.2, and F.6, respectively. We also provide the LLM-as-a-Judge protocol and prompt in Appendix F.3 and the response-generation prompts in Appendix G. These materials specify the main components and experimental settings needed to reproduce our method and evaluation.

REFERENCES

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

Malika Aubakirova, Alex Atallah, Chris Clark, Justin Summerville, and Anjney Midha. State of ai: An empirical 100 trillion token study with openrouter. *arXiv preprint arXiv:2601.10088*, 2026.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3829–3846, 2023.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

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.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, Haofen Wang, et al. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1):32, 2023.

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

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. From rag to memory: Non-parametric continual learning for large language models. *arXiv preprint arXiv:2502.14802*, 2025.

Chuanrui Hu, Xingze Gao, Zuyi Zhou, Dannong Xu, Yi Bai, Xintong Li, Hui Zhang, Tong Li, Chong Zhang, Lidong Bing, et al. Evermemos: A self-organizing memory operating system for structured long-horizon reasoning. *arXiv preprint arXiv:2601.02163*, 2026.

Mengkang Hu, Tianxing Chen, Qiguang Chen, Yao Mu, Wenqi Shao, and Ping Luo. Hiagent: Hierarchical working memory management for solving long-horizon agent tasks with large language model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32779–32798, 2025a.

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

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*, 2021.

Bowen Jiang, Zhuoqun Hao, Young-Min Cho, Bryan Li, Yuan Yuan, Sihao Chen, Lyle Ungar, Camillo J Taylor, and Dan Roth. Know me, respond to me: Benchmarking llms for dynamic user profiling and personalized responses at scale. *arXiv preprint arXiv:2504.14225*, 2025.

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.

Xiaoxi Li, Wenxiang Jiao, Jiarui Jin, Guanting Dong, Jiajie Jin, Yinuo Wang, Hao Wang, Yutao Zhu, Ji-Rong Wen, Yuan Lu, et al. Deepagent: A general reasoning agent with scalable toolsets. In *Proceedings of the ACM Web Conference 2026*, pp. 2219–2230, 2026.

Zhiyu Li, Chenyang Xi, Chunyu Li, Ding Chen, Boyu Chen, Shichao Song, Simin Niu, Hanyu Wang, Jiawei Yang, Chen Tang, et al. Memos: A memory os for ai system. *arXiv preprint arXiv:2507.03724*, 2025.

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.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

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.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*, 2021.

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T Le, Samira Daruki, Xiangru Tang, et al. Reasoningbank: Scaling agent self-evolving with reasoning memory. *arXiv preprint arXiv:2509.25140*, 2025.

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.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Xufang Luo, Hao Cheng, Dongsheng Li, Yuqing Yang, Chin-Yew Lin, H Vicky Zhao, Lili Qiu, et al. On memory construction and retrieval for personalized conversational agents. *arXiv preprint arXiv:2502.05589*, 2025.

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.

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*. Association for Computational Linguistics, 11 2019. URL http://arxiv.org/abs/1908.10084.

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

Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. Agentic retrieval-augmented generation: A survey on agentic rag. *arXiv preprint arXiv:2501.09136*, 2025.

Weiwei Sun, Miao Lu, Zhan Ling, Kang Liu, Xuesong Yao, Yiming Yang, and Jiecao Chen. Scaling long-horizon llm agent via context-folding. *arXiv preprint arXiv:2510.11967*, 2025.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Peng Wang, Zexi Li, Ningyu Zhang, Ziwen Xu, Yunzhi Yao, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. Wise: Rethinking the knowledge memory for lifelong model editing of large language models. *Advances in Neural Information Processing Systems*, 37:53764–53797, 2024a.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuan-Jing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. K-adapter: Infusing knowledge into pre-trained models with adapters. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 1405–1418, 2021.

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*, 2024b.

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.

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

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, et al. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2526–2547, 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.

Xixi Wu, Kuan Li, Yida Zhao, Liwen Zhang, Litu Ou, Huifeng Yin, Zhongwang Zhang, Xinmiao Yu, Dingchu Zhang, Yong Jiang, et al. Resum: Unlocking long-horizon search intelligence via context summarization. *arXiv preprint arXiv:2509.13313*, 2025.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. C-pack: Packaged resources to advance general chinese embedding, 2023.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*, 2025.

Rui Ye, Zhongwang Zhang, Kuan Li, Huifeng Yin, Zhengwei Tao, Yida Zhao, Liangcai Su, Liwen Zhang, Zile Qiao, Xinyu Wang, et al. Agentfold: Long-horizon web agents with proactive context management. *arXiv preprint arXiv:2510.24699*, 2025.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, et al. Memagent: Reshaping long-context llm with multi-conv rl-based memory agent. *arXiv preprint arXiv:2507.02259*, 2025.

Guibin Zhang, Muxin Fu, and Shuicheng Yan. Memgen: Weaving generative latent memory for self-evolving agents. *arXiv preprint arXiv:2509.24704*, 2025a.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025b.

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, 2025c.

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

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*, volume 38, pp. 19724–19731, 2024.

A Limitations and Future Work

A.1 Limitations

While `Threader` demonstrates strong performance and efficiency in long-horizon conversational settings, we identify several potential limitations.

Dependence on segmentation quality. Our approach relies on topic-coherent segmentation to organize raw interactions into retrieval units. Although the proposed segmenter is lightweight and effective in practice, segmentation errors may propagate to downstream retrieval. In particular, if semantically related evidence is split across segment boundaries, it may reduce recall. Designing more robust segmentation strategies or incorporating boundary-aware retrieval remains an important direction for future work.

Scalability under extremely large memory. While our indexing and retrieval pipeline is efficient, scaling to extremely large or lifelong interaction histories may introduce additional storage and retrieval challenges. Incorporating memory pruning, compression, or hierarchical indexing mechanisms could further improve scalability.

Limited abstraction of long-range dependencies. `Threader` emphasizes preserving fine-grained evidence and deferring abstraction to query time. However, some tasks may benefit from higher-level abstraction or long-range reasoning that integrates information across distant segments. Extending the framework to support adaptive abstraction—combining raw evidence with learned summaries—could improve performance in such scenarios.

A.2 Future Work

In future work, we will extend `Threader` along two main directions. First, we will broaden the current factual-memory framework toward more general agent memory scenarios, especially experiential memory for action-oriented agents. Beyond preserving and retrieving user-provided facts in long-horizon dialogue, future extensions will consider task trajectories, action outcomes, tool-use traces, failure cases, and reusable execution strategies as memory units. This requires adapting `Threader`'s topic-coherent segmentation, multi-view representation, and multi-signal retrieval to heterogeneous agent experiences that involve observations, actions, tool outputs, and environmental feedback. Such an extension would allow `Threader` to support not only evidence recall in personalized dialogue, but also long-horizon planning and action execution in tool-use, embodied, web, and coding agents. Second, we will work toward stronger memory governance for privacy protection, user control, and responsible memory management. Important directions include user-facing mechanisms for memory inspection, editing, deletion, and selective forgetting, as well as privacy-preserving storage, access control, retention policies, and safeguards against unauthorized profiling or inappropriate memory use. These extensions aim to make `Threader` more general, controllable, and trustworthy for real-world long-term agent deployments.

B Additional Related Works

B.1 Experiential Memory in Agent Learning

Experiential memory aims to improve an agent's future task-solving ability by storing and reusing past interaction experience, such as trajectories, success/failure cases, skills, and high-level strategies. Early reflective agents use verbal feedback from failed attempts to improve subsequent trials, enabling within-task self-improvement through accumulated reflections (Shinn et al., 2023). Beyond single-task retry, ExpeL stores past trajectories and extracts cross-task insights from successful and failed experiences, allowing agents to retrieve similar experiences or follow distilled strategies at inference time (Zhao et al., 2024). Other systems further transform experience into executable skills, workflows, or reusable policies for open-ended environments and tool-use tasks (Wang et al., 2023; 2024c; Ouyang et al., 2025). These methods are effective for action-oriented agents, where memory primarily supports planning, exploration, and policy improvement. In contrast, our work focuses on factual memory in long-horizon human–assistant dialogue, where the key challenge is not to learn reusable action strategies, but to preserve and retrieve fine-grained user-specific evidence under long, entangled interaction histories.

B.2 Working Memory in Task Execution

Working memory maintains transient task state and intermediate context during ongoing reasoning or execution. Unlike experiential memory, which accumulates reusable knowledge across episodes, working memory is usually invoked within a single task trajectory to manage limited context budgets, preserve important intermediate information, and prevent goal drift. Recent work explores memory mechanisms for long-horizon task execution, including summarizing or folding interaction traces into compact state representations, selectively retaining task-relevant evidence, and organizing subgoals or intermediate artifacts for later reasoning steps (Yu et al., 2025; Wu et al., 2025; Hu et al., 2025a). Such mechanisms are particularly useful for web navigation, deep research, tool-use, and multi-step reasoning, where the agent must continuously update its internal workspace as observations and

actions accumulate (Sun et al., 2025; Ye et al., 2025; Li et al., 2026). However, working memory is typically optimized for maintaining task progress within an active execution process, rather than building a persistent, queryable record of long-term user history. Our setting instead requires a durable memory bank that supports future factual recall across sessions, making evidence preservation and retrieval fidelity central design concerns.

B.3 Parametric and Latent Memory.

Agent memory can also be realized in parametric or latent forms, beyond explicit token-level storage (Hu et al., 2025b). Parametric memory stores information in learnable parameters, either by editing the backbone or by attaching additional parameter modules. Representative model-editing methods, such as MEND (Mitchell et al., 2021) and ROME (Meng et al., 2022), inject factual knowledge through localized parameter updates, while adapter-style approaches such as K-Adapter (Wang et al., 2021) store additional knowledge in modular parameters. Latent memory instead stores information in continuous internal representations, such as hidden states, memory tokens, KV caches, or compressed embeddings. For example, MemoryLLM and its extension M+ introduce persistent memory tokens or cross-layer token pools in the latent space of the transformer for self-updatable long-term knowledge storage (Wang et al., 2024b; 2025), while AutoCompressor compresses long contexts into compact summary vectors for later reuse (Chevalier et al., 2023).

Although parametric and latent memory can compactly internalize knowledge and reduce explicit retrieval overhead, they are less aligned with our target setting of long-term personalized dialogue. Parametric updates are often model-dependent, costly to revise, and vulnerable to interference or catastrophic forgetting, while latent memory is difficult to inspect, edit, or attribute to concrete dialogue evidence. `Threader` therefore adopts token-level factual memory: instead of internalizing dialogue history into parameters or hidden states, it preserves raw dialogue content as externally accessible topic-coherent evidence segments, prioritizing transparency, controllability, and evidence fidelity.

C Algorithm

The overall procedure of `Threader` is summarized in Algorithm 1.

D Datasets

We provide detailed descriptions of the two long-term dialogue memory benchmarks used in our evaluation: LongMemEval (Wu et al., 2024) and PersonaMem (Jiang et al., 2025). Both datasets are designed to evaluate whether an agent can retrieve and use information from long interaction histories, but they emphasize different aspects of memory. LongMemEval focuses on multi-session factual and temporal memory over user–assistant conversations, while PersonaMem stresses long-session personalization, preference understanding, and user-specific response generation. The statistics of the two datasets are summarized in Table 5.

LongMemEval. LongMemEval is a long-term memory benchmark for evaluating whether dialogue agents can answer questions grounded in previous user–assistant interactions. Each test instance consists of a user query, a long multi-session conversation history, and a gold answer. The model is required to identify the relevant evidence from the historical dialogue and generate an answer

Table 5: **Dataset statistics.** We report the number of QA pairs, the number of samples, and dialogue-level statistics for each benchmark. All metrics with "/" denote average values, including sessions per sample, tokens per session, turns per session, and tokens per turn.

Dataset	# QA	# Sample	Sess./Sample	Tok./Sess.	Turn/Sess.	Tok./Turn
LongMemEval	500	500	47.7	2,160.73	5.12	421.88
PersonaMem	589	37	1	26,165.38	174.65	149.82

consistent with the gold response. Compared with standard open-domain QA, LongMemEval places stronger emphasis on memory retrieval, since the information needed to answer the question is usually contained in past conversations rather than in the model's parametric knowledge.

LongMemEval contains 500 QA instances, with an average session length of approximately 2,160 tokens. Its questions cover several question types that assess different long-term memory abilities, including single-session user information recall, single-session assistant information recall, preference recall, multi-session reasoning, knowledge update, and temporal reasoning. These question types make LongMemEval suitable for evaluating whether a memory system can locate sparse evidence, distinguish outdated and updated information, and reason over events mentioned across different sessions.

Algorithm 1: `Threader`

Input: $C = \{S_1, \ldots, S_n\}$: conversation history; $S_i = (t_1, \ldots, t_m)$: the i-th session; $t_j = (u_j, a_j)$: the j-th user-assistant turn; q: the user query; g_θ: the incremental topic-coherent segmenter; f_{enc}: the encoder; $\text{BM25}(\cdot,\cdot)$: the lexical retriever; $\text{Rerank}(\cdot,\cdot)$: the cross-encoder reranker.
Output: The final generated response for the given user query q.

// Incremental Topic-Coherent Segmentation
Construct topic-coherent evidence segments from the conversation history C
foreach *session* $S_i = (t_1, \ldots, t_m)$ *in* C **do**
  Initialize the current segment $s' \leftarrow (t_1)$ and the segment set $\mathcal{S}_i \leftarrow \emptyset$
  foreach *incoming turn* t_{j+1} *in* S_i **do**
    Construct the segmentation input by concatenating the current segment s' and the incoming turn t_{j+1}
    Predict whether t_{j+1} should continue the current segment or start a new segment using g_θ
    if *the prediction is* CONTINUE **then**
      Append t_{j+1} to the current segment s'
    else
      Add s' to $\mathcal{S}_i$ and start a new segment with t_{j+1}
  Add the final segment s' to $\mathcal{S}_i$
Obtain the full segment set $\mathcal{S} = \bigcup_i \mathcal{S}_i$
// Multi-View Segment Representation
foreach *topic-coherent segment* $s_i = (t_{i,1}, \ldots, t_{i,m})$ *in* $\mathcal{S}$, *where* $t_{i,j} = (u_{i,j}, a_{i,j})$ **do**
  Construct three complementary views: $x_i^{(u)} = \text{Concat}(u_{i,1}, \ldots, u_{i,m})$, $x_i^{(a)} = \text{Concat}(a_{i,1}, \ldots, a_{i,m})$, and $x_i^{(f)} = \text{Concat}(u_{i,1}, a_{i,1}, \ldots, u_{i,m}, a_{i,m})$
  Encode each view using the shared encoder: $e_i^{(v)} = f_{\text{enc}}(x_i^{(v)}),\ v \in \{u, a, f\}$
  Store $\{e_i^{(u)}, e_i^{(a)}, e_i^{(f)}\}$ for dense retrieval
Build dense indices over the multi-view segment representations and a BM25 index over raw segments
// Stage I: High-Recall Segment Retrieval
Encode the user query q using the shared encoder f_{enc}:$z(q) = f_{\text{enc}}(q)$
Compute the multi-view dense semantic score for each segment via Eq. equation 3
Compute the lexical matching score for each segment
Construct the high-recall candidate set $\mathcal{C}$ by combining dense and lexical retrieval results via Eq. equation 4
// Stage II: Evidence-Level Refinement
foreach *candidate segment* s_i *in* $\mathcal{C}$ **do**
  Compute message-level similarities between q and each user/assistant message in s_i
  Select the top-K_2 message-level similarities and average them to obtain the refined dense score via Eq. equation 5
  Combine the refined dense signal with the lexical matching signal via Eq. equation 6
  Apply the cross-encoder reranker and aggregate it with the hybrid score to obtain the final segment score via Eq. equation 7
Select the top-K segments from $\mathcal{C}$ according to the final segment score
Obtain the retrieved evidence segments $\mathcal{R}$
// Response Generation
Generate the final response by conditioning the response model on the user query q

We briefly describe the question types in LongMemEval as follows.

- **Single-session user information recall** evaluates whether the model can recall facts explicitly mentioned by the user within a past session, such as personal attributes, plans, constraints, or experiences.
- **Single-session assistant information recall** evaluates whether the model can remember information provided by the assistant in previous interactions, such as recommendations, explanations, or decisions made during the dialogue.
- **Single-session preference recall** evaluates whether the model can identify user preferences expressed in prior conversations, including likes, dislikes, habits, or requirements.
- **Multi-session reasoning** requires the model to combine evidence distributed across multiple sessions. This question type tests whether the memory system can retrieve complementary pieces of information rather than relying on a single isolated turn.
- **Knowledge update** evaluates whether the model can handle evolving information. The model must identify the most recent or valid state when earlier information is later revised, corrected, or superseded.
- **Temporal reasoning** requires the model to reason about time-sensitive information, such as event order, relative dates, or changes across sessions.

PersonaMem. PersonaMem is a long-session personalization benchmark designed to evaluate whether a dialogue agent can understand and use user-specific information over extended interactions. Each instance contains a long user–assistant dialogue history and a question that requires the model to recall, infer, or apply information about the user. Compared with LongMemEval, PersonaMem contains substantially longer sessions, with an average session length of approximately 26,165 tokens, making it especially suitable for testing memory systems under long-context and high-entropy interaction settings.

PersonaMem contains 589 QA instances and covers seven question types. These question types evaluate not only direct fact recall, but also the model's ability to generalize user preferences, explain preference changes, and make personalized suggestions. This makes PersonaMem a challenging benchmark for long-term user modeling, since relevant evidence is often sparse, interleaved with unrelated topics, and distributed across a long interaction history.

We describe the seven question types in PersonaMem as follows.

- **Generalizing to new scenarios** evaluates whether the model can apply previously observed user preferences or behaviors to a new situation. The answer is not always a direct copy of a past utterance, but should be consistent with the user's established profile.
- **Preference-aligned recommendation** requires the model to provide recommendations that match the user's historical preferences, constraints, or goals. This question type evaluates whether the model can retrieve and synthesize relevant preference evidence.
- **Recalling user-shared facts** evaluates whether the model can recall factual information explicitly shared by the user, such as background, interests, routines, or personal constraints.
- **Recalling facts mentioned by the user** focuses on information that appeared in the user's prior messages, including facts about entities, events, or external situations discussed by the user.
- **Recalling reasons behind previous updates** evaluates whether the model can remember why a preference, plan, or decision changed. This requires recovering not only the updated state, but also the explanatory evidence behind the update.
- **Suggesting new ideas** requires the model to propose new options or actions based on the user's long-term preferences and past interactions. This question type tests whether memory can support personalized generation beyond direct extraction.
- **Tracking full preference evolution** evaluates whether the model can follow how a user's preference changes over time. The model must distinguish earlier preferences from later updates and generate an answer consistent with the complete preference trajectory.

Overall, the two datasets provide complementary evaluation settings. LongMemEval offers a controlled multi-session memory QA benchmark with diverse question types, while PersonaMem presents

longer and more personalized interaction histories that better stress-test memory systems under realistic long-session conditions. Together, they allow us to evaluate both factual memory retrieval and personalized memory utilization in long-horizon user–assistant dialogue.

Although LoCoMo (Maharana et al., 2024) has been widely adopted in prior work for evaluating long-term dialogue memory, we do not include it in our main evaluation because it differs from our target setting in both session length and interaction format. First, each LoCoMo session contains only about 586 tokens on average, making it substantially shorter than the long-session histories considered in this work. Second, LoCoMo is constructed from simulated two-person conversations, whereas our focus is long-term personalized memory for user–assistant interaction histories. We therefore use LongMemEval and PersonaMem as the primary benchmarks, as they better align with the long-horizon human–assistant setting studied in this paper.

E Baseline and Implementation Details

Our baselines cover both persistent memory systems and adaptive retrieval frameworks. For methods with official public implementations, we follow the released codebases and recommended default settings whenever they are compatible with our evaluation pipeline. In particular, several baselines use OpenAI embedding models, such as `text-embedding-3-large` and `text-embedding-ada-002`, as their default embedding backbones; we keep these default configurations and follow the official OpenAI embedding API documentation.[2] For Self-RAG (Asai et al., 2024), since the original release is a specially fine-tuned model, we instead use a LangGraph-based re-implementation of its retrieve-generate-reflect workflow with our shared backbone models, to ensure a fair comparison across methods. All public datasets, pretrained models, and baseline codebases used in this work are credited to their original authors and used under their respective licenses and terms of use.

Self-RAG. Self-RAG is not a persistent memory system in the strict sense, but it serves as a strong adaptive retrieval baseline Asai et al. (2024). It trains the model to decide when retrieval is necessary during generation and to emit reflection tokens that critique retrieval relevance, evidential support, and response utility, thereby forming a retrieve-generate-critique loop. Compared with standard RAG pipelines that retrieve in a fixed manner, Self-RAG improves factual grounding while preserving the general-purpose flexibility of the underlying language model. In our implementation, we use a LangGraph-based re-implementation with GPT-4.1-mini as the general backbone LLM, and instantiate the dense retriever with the `facebook/contriever-msmarco` (Izacard et al., 2021). The retriever encodes both queries and dialogue-history chunks for similarity-based retrieval, and the retrieved chunks are then fed into our unified response model for answer generation.

HippoRAG2. We use HippoRAG2, introduced in *From RAG to Memory*, as a structure-aware retrieval baseline for non-parametric continual learning (Gutiérrez et al., 2025). The method builds an open knowledge graph over the corpus, performs deeper passage integration, and applies query-conditioned *Personalized PageRank* to capture multi-hop associations and relational propagation beyond plain semantic matching. By combining dense retrieval signals, graph-based propagation, and more effective online use of an LLM, HippoRAG2 moves beyond standard vector RAG toward a more organized and continually extensible external memory mechanism. In our implementation, we use the official HippoRAG2 framework with GPT-4.1-mini as the general construction LLM and `paraphrase-multilingual-MiniLM-226 L12-v2` (Reimers & Gurevych, 2019) as embedding model. The retrieved memory chunks are then fed into our unified response model for answer generation.

Mem0. Mem0 models long-term memory as a module decoupled from response generation and maintains an external memory store through incremental extraction, matching, and update operations (Chhikara et al., 2025). At each step, it extracts salient facts from the current interaction, matches them against existing memories, and uses an LLM-based update policy to insert, merge, delete, or discard candidate memories. In our implementation, we use the official Mem0 framework with GPT-4.1-mini as the general memory construction LLM and keep its default embedding model,

[2] https://platform.openai.com/docs/api-reference/embeddings

`text-embedding-3-large`. The retrieved memories are then fed into the shared response LLM specified in Section 4.1 for answer generation.

MemU. MemU is an always-on proactive memory framework designed for long-running AI agents.[3] It organizes memory through a file-system-like hierarchy of *Resources*, *Items*, and *Categories*, where raw inputs are preserved as resources, extracted facts and preferences are stored as items, and topic-level categories support higher-level context assembly. Its `memorize` pipeline continuously ingests interactions, extracts memory items, updates categories, and cross-references existing memories. During retrieval, MemU supports both embedding-based RAG for fast context loading and LLM-based retrieval for deeper intent prediction and query refinement. This design enables agents to maintain structured memory, anticipate user needs, and surface relevant context proactively in long-running settings. In our implementation, we use the official MemU framework with GPT-4.1-mini as the general memory construction LLM and keep its default embedding model, `text-embedding-ada-002`. The retrieved memories are then fed into the shared response LLM specified in Section 4.1 for answer generation.

A-Mem. A-Mem formulates memory management as an agentic note-taking process inspired by the Zettelkasten paradigm Xu et al. (2025). Each interaction is converted into a structured note containing content, timestamps, keywords, tags, contextual descriptions, and embeddings. The system then retrieves related notes and lets an LLM controller decide how to revise existing notes, create links, and evolve the memory graph. Compared with conventional memory stores that mainly support storage and retrieval, A-Mem places greater emphasis on memory organization and self-evolution. In our implementation, we use the official A-Mem framework with GPT-4.1-mini as the general memory construction LLM and keep its default embedding model, `all-MiniLM-L6-v2` (Reimers & Gurevych, 2019). The retrieved memories are then fed into the shared response LLM specified in Section 4.1 for answer generation.

MemOS. MemOS treats memory as a first-class system resource and introduces the unified *MemCube* abstraction over plaintext, activation-based, and parameter-level memory (Li et al., 2025). Rather than reducing memory to a single vector store, it coordinates memory writing, maintenance, loading, and retrieval through layered scheduling and lifecycle management mechanisms. This OS-style design is intended to support long-horizon reasoning, continual personalization, and reusable knowledge across tasks and environments within a single memory framework. In our implementation, we use the official MemOS framework and set GPT-4.1-mini as the general model. Since MemOS provides a high-level system interface with limited access to its internal scheduling details, we keep its internal memory scheduling and retrieval configuration unchanged and only standardize the external LLM backbone for fair comparison.

EverMemOS. EverMemOS extends the operating-system view of memory by emphasizing self-organization for structured long-horizon reasoning Hu et al. (2026). It first converts interaction streams into structured *MemCells* that capture episodic traces, atomic facts, and short-term foresight signals, and then consolidates them into higher-level *MemScenes* that encode themes, user profiles, and stable semantic structure. During inference, EverMemOS performs reconstructive recollection, assembling task-relevant context from structured long-term memory rather than simply retrieving a few isolated fragments. This design makes it particularly suitable for scenarios where memory must remain coherent across extended interactions. In our implementation, we use the official EverMemOS framework with GPT-4.1-mini as the general memory construction LLM and keep its default embedding model, `Qwen3-Embedding-4B` (Zhang et al., 2025b) and reranking model, `Qwen3-Reranker-4B` (Zhang et al., 2025b). The retrieved memories are then fed into the shared response LLM specified in Section 4.1 for answer generation.

`Threader`. `Threader` follows a lightweight, non-LLM memory construction design. Specifically, it uses `paraphrase-multilingual-MiniLM-L12-v2` (Reimers & Gurevych, 2019) as the embedding encoder and `bge-reranker-base` (Xiao et al., 2023) as the cross-encoder reranker. Both models are lightweight off-the-shelf encoders and do not require LLM calls during memory construction. Inputs exceeding the encoder context window are handled by the model's

[3] https://github.com/NevaMind-AI/memU

Table 6: **Statistics of the constructed motivation data.** We progressively concatenate adjacent sessions in LongMemEval to create longer-session settings. The number of QA examples and the total dialogue content are kept unchanged, while the average number of sessions per sample decreases and the average session length increases.

Name	Setting	Sessions / Sample	Tokens / Session
LME	Original	47.65	2,311.21
LME-M2	2 session concatenation	23.55	4,676.53
LME-M4	4 session concatenation	11.50	9,576.78
LME-M8	8 session concatenation	5.48	20,085.12

default truncation behavior. For longer real-world text units, `Threader` can be instantiated with longer-context embedding or reranking models without changing the framework.

F Additional Experimental Results and Analysis

F.1 Detailed Memory Fidelity Analysis of Mem0 and MemU

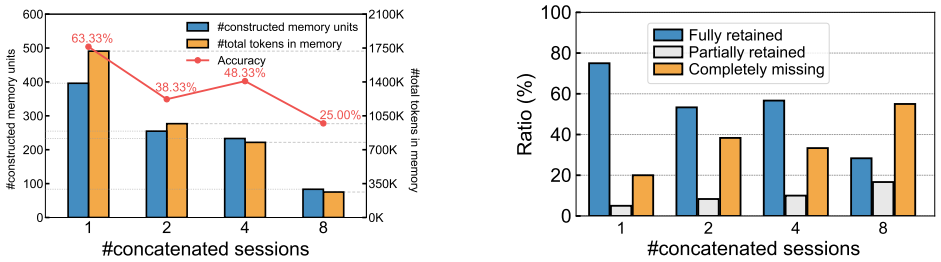
We construct the motivation dataset from LongMemEval by sampling 60 instances, with 10 instances from each question type. For each sample, we keep the original QA pairs, evidence annotations, and total dialogue content unchanged, and only modify the session granularity by concatenating adjacent sessions. This yields four settings: the original data, $2\times$, $4\times$, and $8\times$ session concatenation. As shown in Table 6, this process progressively reduces the average number of sessions per sample from 47.65 to 5.48, while increasing the average session length from 2,311.21 to 20,085.12 tokens. We then evaluate two representative rewrite-based memory methods, Mem0 and MemU, on these motivation datasets to examine how LLM-based memory construction behaves as individual sessions become longer and more information-dense. Both methods perform memory construction at the session level rather than the turn level, making them directly affected by changes in session granularity.

The main results are summarized in Section 1. As shown in Fig. 1a and Fig. 4a, when sessions become longer after concatenation, both Mem0 and MemU construct fewer and less informative memory entries, indicating that the memory writing process becomes increasingly lossy. Since these methods construct memory by directly rewriting each session into memory entries, longer sessions force the LLM-based memory writer to process more heterogeneous information within a single input. This makes query-relevant facts, especially those located in the middle of long sessions or surrounded by unrelated topics, more likely to be compressed away or omitted, consistent with the well-known lost-in-the-middle behavior of long-context LLMs (Liu et al., 2024). This degradation further affects downstream QA accuracy, which drops as the session length increases. We also observe that the degradation is not smooth: QA correctness fluctuates across different concatenation settings, suggesting that rewrite-based memory construction is unstable under long and high-entropy sessions.

To diagnose this issue, we further examine whether the annotated evidence is preserved in the constructed memory banks. Specifically, we manually inspect the memory entries produced by Mem0 and MemU for each QA instance and compare them against the annotated evidence required to answer the question. If all required evidence is fully preserved in the memory bank, we label the instance as *Fully Retained*. If none of the required evidence is preserved, we label it as *Completely Missing*. For QA instances that require multiple pieces of evidence, if only a subset of the required evidence is retained, we label the instance as *Partially Retained*. Fig. 1b and Fig. 4b show that a substantial portion of query-relevant evidence is only partially retained or completely missing, confirming that many failures originate from the memory construction stage rather than the final response generation alone.

Based on the above results, we identify two characteristic failure patterns of rewrite-based memory construction under increasingly long sessions.

Phenomenon 1: Memory degradation. As individual sessions become longer and more information-dense, the constructed memory becomes progressively smaller and less informative. In other words, the memory system stores fewer memory items and fewer words, even though the original dialogue



(a) Accuracy and memory size of MemU.

(b) Evidence preservation in MemU.

Figure 4: Limitations of existing LLM-based memory construction methods (MemU). As interaction length and information entanglement increase (simulated via session concatenation with LongMemEval), MemU exhibits (a) degraded QA accuracy and shrinking memory size, (b) increasing loss of query-relevant evidence.

contains the same amount of information. This indicates that the memory writer increasingly compresses, filters, or omits details when processing long and heterogeneous sessions, causing query-relevant evidence to disappear from the memory bank.

Phenomenon 2: Uncontrolled memory retention. Memory retention is not monotonic or controllable across difficulty settings. Evidence retained in a harder setting is not necessarily retained in an easier setting, and vice versa. This suggests that rewrite-based memory construction is sensitive to session boundaries and local context organization: whether a specific fact is preserved depends not only on its importance, but also on how it is surrounded by other information during memory writing.

Case Study. We provide two case studies to illustrate the above phenomena. Figure 5 illustrates **Phenomenon 1**. In Sample 074 and Sample 098, as the session concatenation factor increases from the original LongMemEval setting to the 8× setting, both Mem0 and MemU retain fewer memory items and fewer words. More importantly, the retained memories gradually lose the evidence required for answering the query. For example, in Sample 074, the answer requires aggregating two camping trips, i.e., a 3-day trip to Big Sur and a 5-day trip to Yellowstone. However, under longer-session settings, the constructed memory increasingly misses one of these two pieces of evidence, making it impossible to recover the correct answer. Similarly, in Sample 098, the answer requires comparing the accommodation costs in Maui and Tokyo, but the relevant evidence becomes partially retained or completely missing as sessions become longer. Here, MI denotes the number of constructed memory items, and W denotes the total number of words in the constructed memory. The consistent reduction of MI and W shows that long-session construction causes memory degradation rather than merely retrieval failure.

Figure 6 illustrates **Phenomenon 2**. The examples show that memory retention is unstable across concatenation settings. In Sample 304, the evidence that the user bought a smoker is retained in some harder settings but missing in easier ones. In Sample 340, evidence about the skin tag removal and persistent cough also appears inconsistently across settings and methods. This demonstrates that rewrite-based memory construction does not provide a controllable guarantee on evidence preservation. The three emoji markers indicate the evidence status: *Covered*, *Partial*, and *Missing*. Even when the same dialogue content and QA target are used, changing only the session granularity can lead the memory writer to preserve, partially preserve, or completely omit the same evidence.

F.2 BERT Segmenter Training Details

F.2.1 BERT Segmenter Training Settings

Task formulation. We formulate incremental topic segmentation as a binary classification problem. Given the current segment buffer s' and an incoming turn t', the segmenter predicts whether to append t' to the current segment (`CONTINUE`) or start a new segment (`SEGMENT`). Each training example therefore corresponds to a local boundary decision, matching the sequential decision process used during inference. Segmentation never splits a sentence across segments.

Sample	074		098	
Question	How many days did I spend on camping trips in the United States this year?		How much more did I spend on accommodations per night in Hawaii compared to Tokyo?	
Answer	8 days		$270	
Method	Mem0	MemU	Mem0	MemU
LME	**MI=162 W=13536** 1. "Just got back from a ***3-day solo camping trip*** to Big Sur in early April 2023." 2. "Just got back from a ***5-day camping trip*** to Yellowstone National Park in March 2023." ✓	**MI=478 W=168279** 1. They recently returned from a "***5-day camping trip*** to Yellowstone National Park last month" and ... 2. User011 recently returned from a ***3-day solo camping trip*** to Big Sur in early April, where ... ✓	**MI=176 W=15094** 1. "Stayed in a hostel ***in Tokyo that cost around $30 per night*** when ..." 2. "Booked a luxurious resort ***in Maui that costs over $300 per night***." ✓	**MI=431 W=149665** 1. They have already booked a luxurious resort ***in Maui that costs over $300 per night*** and ... 2. They mentioned having stayed in a hostel ***in Tokyo that cost around $30 per night*** during ... ✓
LME-M2	**MI=97 W=7975** 1. "Just got back from a ***5-day camping trip*** to Yellowstone National Park on 2023/03/29." 2. "User recently returned from a ***3-day solo camping trip*** to Big Sur in early April 2023." ✓	**MI=406 W=121111** 1. They recently returned from an "amazing" ***5-day camping trip*** to ... 2. User011 recently returned from a ***3-day solo camping trip*** to Big Sur in early April. ✓	**MI=110 W=8634** 1. "Booked a luxurious resort ***in Maui that costs over $300 per night***." ✗	**MI=305 W=93009** 1. User014 planned a trip to ***Maui*** and ... costing ***over $300 per night*** and ... 2. User014 shared that they previously stayed in a hostel ***in Tokyo that cost around $30 per night*** during a solo trip in January. ✓
LME-M4	**MI=59 W=5584** 1. "User recently went on a ***3-day solo camping trip*** to Big Sur in early April 2023." ✗	**MI=324 W=86753** 1. User011 recently returned from a ***3-day solo camping trip*** to ... 2. They ... an amazing ***5-day camping trip*** to Yellowstone National Park last month. ✓	**MI=44 W=3165** (Failed to retain the relevant memory) ✗	**MI=183 W=47298** 1. User014 planned a trip to ***Maui*** and ... they had booked a luxurious resort costing ***over $300 per night*** and ... 2. User014 ... in a hostel ***in Tokyo for around $30 per night*** during a solo trip in January. ✓
LME-M8	**MI=28 W=2202** 1. "User recently got back from a ***3-day solo camping trip*** to Big Sur in early April." ✗	**MI=18 W=29203** 1. User011 completed a ***5-day camping trip*** to Yellowstone National Park last month and is still buzzing from the experience. ✗	**MI=20 W=1145** (Failed to retain the relevant memory) ✗	**MI=23 W=22746** (Failed to retain the relevant memory) ✗

Figure 5: **Case study of memory degradation.** As the concatenation factor increases, Mem0 and MemU construct fewer memory items and fewer words, and progressively lose query-relevant evidence. MI denotes the number of memory items, and W denotes the total number of words in the constructed memory.

Training data construction. We construct segmentation supervision from the dialogue histories of abstention instances in LongMemEval. Although these instances have unanswerable questions, their histories contain multi-session user–assistant interactions with natural session boundaries and intra-session topic transitions. We use only the dialogue histories and their boundary annotations; the associated questions and answers are not used as training targets. All non-abstention evaluation instances are excluded from segmenter training.

Following Section 3.1, we combine two sources of boundary supervision. Session boundaries provide coarse-grained hard labels, while GPT-4.1-mini annotates finer-grained topic boundaries within each

Sample	304		340	
Question	What kitchen appliance did I buy 10 days ago?		Which health issue did I deal with first, the persistent cough or the skin tag removal?	
Answer	A smoker		Persistent cough	
Method	Mem0	MemU	Mem0	MemU
LME	**MI=212 W=18639**	**MI=431 W=149665**	**MI=192 W=16694**	**MI=31 W=14968**
	1. "Got ***a smoker*** on 2023/03/15 and excited to experiment with different types of wood and meats." 😃	(Failed to retain the relevant memory) 😕	1. "Recently had a ***skin tag*** removed on ***2023/02/22***." 2. "User has been experiencing a ***persistent cough*** ... a visit to their primary care physician, Dr. Smith, on ***2023/02/10***." 😃	(Failed to retain the relevant memory) 😕
LME-M2	**MI=78 W=6806**	**MI=305 W=93009**	**MI=97 W=8575**	**MI=421 W=126071**
	(Failed to retain the relevant memory) 😕	(Failed to retain the relevant memory) 😕	(Failed to retain the relevant memory) 😕	1. They ... to remove a ***skin tag*** on their neck on ***February 22nd***, which ... 😐
LME-M4	**MI=54 W=5112**	**MI=44 W=47298**	**MI=48 W=4355**	**MI=28 W=76486**
	1. "User got ***a smoker*** on 2023/03/15 and is experimenting with different types of wood and meats." 😃	1. (2023/03/15) User036 ... and meats after recently acquiring ***a smoker***. 2. (2023/03/15) User036 shared their excitement about experimenting with ***a new smoker*** and ... 😃	1. "User had a minor surgical procedure to remove a ***skin tag*** on their neck on ***2023/02/22***, and it has healed up now." 😐	1. User038 ... remove a ***skin tag*** from their neck on ***February 22nd***, which ... 2. User038 has been dealing with a ***persistent cough*** ... primary care physician, Dr. Smith, ***on February 10th***. 😃
LME-M8	**MI=29 W=2365**	**MI=23 W=22746**	**MI=31 W=3104**	**MI=23 W=36654**
	(Failed to retain the relevant memory) 😕	1. (2023/03/15) User036 seeks new BBQ sauce recipes and mentions they just got ***a smoker***, excited ... 😃	1. "Had a minor surgical procedure to remove a ***skin tag*** on ***2023/02/22***" 😐	1. Additionally, they had a minor surgical procedure to remove a ***skin tag*** from their neck on ***February 22nd***, and 😐

Figure 6: **Case study of uncontrolled memory retention.** Rewrite-based memory construction does not preserve evidence monotonically across session-granularity settings. The same evidence may be covered, partially retained, or missing under different concatenation factors. The emoji markers denote *Covered*, *Partial*, and *Missing*, respectively. MI denotes the number of memory items, and W denotes the total number of words in the constructed memory.

session. These LLM-derived annotations provide complementary weak supervision for detecting topic shifts that do not coincide with session boundaries. Both sources are converted into binary `CONTINUE`/`SEGMENT` targets.

Importantly, the number of source benchmark instances is different from the number of supervised training examples. Each source instance contains multiple sessions, and each annotated session yields a sequence of boundary-decision examples. For a continuation example, the input consists of the

preceding segment context and the next turn within the same segment; for a boundary example, it consists of the preceding segment and the first turn of the next segment. Although the supervision is derived from 30 source instances, this construction yields **15,260 supervised boundary-decision examples**, containing approximately 13M WordPiece tokens across the constructed inputs. The segmenter therefore learns from thousands of local boundary decisions despite the small number of source instances.

We partition the source histories at the dialogue level, assigning 90% to training and 10% to validation, so that examples derived from the same source history remain in the same split. The validation set is used for checkpoint selection and comparison of candidate encoder backbones. The original class distribution is approximately `CONTINUE` : `SEGMENT` $= 4 : 1$, reflecting the greater frequency of topic continuation in the annotated conversations.

Encoder backbones and training protocol. We compare the base-size variants of BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), and ModernBERT (Warner et al., 2025) under the same binary classification formulation. For each backbone, we attach a linear classification head to the final representation of the classification token and fine-tune the entire model.

Let $y_i \in \{0, 1\}$ denote the target label for example i, where 1 corresponds to `SEGMENT`, and let p_i denote the predicted boundary probability. We optimize the standard binary cross-entropy loss:

$$\mathcal{L}_{\mathrm{CE}} = -\frac{1}{N} \sum_{i=1}^{N} \left[y_i \log p_i + (1 - y_i) \log(1 - p_i) \right]. \tag{8}$$

All backbones are trained for 5 epochs using AdamW, with a batch size of 8, a learning rate of 2×10^{-5}, and FP16 mixed precision on a single NVIDIA RTX 3090 GPU.

Evaluation protocol and metrics. We assess segmentation quality on a subset of 50 LongMemEval instances disjoint from the segmenter training data. Reference boundaries are constructed using the same annotation protocol as for training, combining session boundaries with LLM-annotated intra-session topic shifts. This subset also serves to calibrate the boundary decision threshold. Calibration uses only boundary annotations and segmentation metrics, without QA labels, gold answers, response predictions, or downstream QA accuracy.

Boundary detection accuracy alone does not fully characterize the usefulness of segmentation for memory retrieval. Excessive segmentation can fragment coherent evidence, whereas insufficient segmentation can mix unrelated topics. We therefore evaluate both boundary accuracy and segment granularity using five complementary metrics.

Let $\mathcal{D}$ denote the evaluated dialogues. For dialogue d, let T_d be its number of turns, P_d and G_d its predicted and reference boundary sets, and $\mathcal{S}_d^{\mathrm{pred}}$ and $\mathcal{S}_d^{\mathrm{gold}}$ the corresponding segment partitions.

- **Exact-match F1** measures agreement between predicted and reference boundaries. We compute precision and recall as

$$\mathrm{Prec}_d = \frac{|P_d \cap G_d|}{|P_d|}, \qquad \mathrm{Rec}_d = \frac{|P_d \cap G_d|}{|G_d|}, \tag{9}$$

  and macro-average the resulting dialogue-level F1 scores.
- **Window-tolerant F1 (W-F1)** allows a boundary offset of one turn. Its precision is the fraction of predicted boundaries within one turn of a reference boundary, and its recall is the fraction of reference boundaries within one turn of a predicted boundary. We report their harmonic mean.
- **Boundary Over-segmentation Ratio (BOR)** measures the number of predicted boundaries relative to the reference:

$$\mathrm{BOR} = \frac{\sum_{d \in \mathcal{D}} |P_d|}{\sum_{d \in \mathcal{D}} |G_d|}. \tag{10}$$

  Values above or below 1 indicate over-segmentation or under-segmentation, respectively. A value close to 1 indicates agreement in boundary count, but does not by itself establish boundary accuracy.

Table 7: Performance comparison of varying architectures and thresholds on a randomly sampled subset of 50 evaluation instances from the LongMemEval dataset. The arrows (↑, → 1) indicate the optimal direction for each metric. Best and second-best results **within each model family** are highlighted in **bold** and underlined, respectively. The highlighted row indicates our final selected segmenter.

Base Model	Threshold (τ)	LongMemEval Dataset (50 Samples)				
		F_1↑	W-F_1↑	BOR→ 1	Pur.↑	Cov.↑
BERT Family						
BERT	0.15	**0.497**	**0.600**	1.266	**0.777**	0.682
BERT	**0.20**	0.483	0.594	1.169	0.763	0.708
BERT	0.25	0.475	0.592	**1.100**	0.754	**0.727**
RoBERTa Family						
RoBERTa	0.15	0.442	**0.658**	1.754	**0.852**	0.550
RoBERTa	0.20	0.463	0.653	1.630	0.836	0.585
RoBERTa	0.25	**0.479**	0.645	**1.502**	0.819	**0.619**
ModernBERT Family						
ModernBERT	0.30	**0.482**	**0.600**	1.093	**0.774**	0.697
ModernBERT	0.35	0.478	0.597	1.056	0.768	0.709
ModernBERT	0.40	0.470	0.594	**1.023**	0.762	**0.719**

- **Purity** measures how well each predicted segment is contained within a reference segment:

$$\text{Purity}_d = \frac{1}{T_d} \sum_{p \in \mathcal{S}_d^{\text{pred}}} \max_{g \in \mathcal{S}_d^{\text{gold}}} |p \cap g|. \tag{11}$$

- **Coverage** measures how well each reference segment is preserved within a predicted segment:

$$\text{Coverage}_d = \frac{1}{T_d} \sum_{g \in \mathcal{S}_d^{\text{gold}}} \max_{p \in \mathcal{S}_d^{\text{pred}}} |g \cap p|. \tag{12}$$

Backbone and threshold selection. Table 7 compares the candidate backbones at representative decision thresholds. We select **BERT-base with** $\tau = 0.20$ as the default segmenter, considering boundary accuracy together with segment coherence and completeness. Within the BERT family, lowering the threshold to 0.15 improves exact-match F1 but also increases over-segmentation and reduces Coverage. Raising it to 0.25 improves boundary-count calibration and Coverage at the expense of F1 and Purity. The selected threshold provides a practical balance between these competing properties.

The selected model and threshold are fixed for all downstream experiments. In particular, we directly apply the same segmenter to PersonaMem without dataset-specific fine-tuning or threshold adjustment.

F.2.2 Controlled Training Settings

We further analyze the effects of training data size, class distribution, and supervision mixture. Across all experiments, training hyperparameters are fixed to the default values described above, including the learning rate, batch size, number of epochs, and input-context configuration.

To obtain additional examples for the different data configurations, we supplement the candidate training pool with examples constructed from non-abstention LongMemEval dialogue histories. These histories undergo the same annotation and example-construction pipeline as the original data: session boundaries provide hard supervision, GPT-4.1-mini annotates finer-grained intra-session boundaries, and the annotated histories are converted into binary boundary-decision examples.

Table 8: Effect of training data size on segmentation quality. Training sizes are expressed relative to the original setting. The original configuration is shown in bold.

Relative training size	F1↑	BOR→ 1
50%	0.399	0.914
80%	0.471	1.114
90%	0.454	1.054
100% (original)	**0.483**	**1.169**
110%	0.486	1.247
120%	0.478	1.205

The training-data-size analysis varies the total number of training examples. For the class-distribution and supervision-mixture analyses, we construct training sets with the specified ratios from the candidate pool while keeping the total number of training examples identical across settings within each analysis.

Effect of training data size. To examine whether the constructed supervision is sufficient, we compare training sizes ranging from 50% to 120% of the original setting. For the 110% and 120% settings, we expand the training data with additional dialogue histories from LongMemEval. These histories undergo the same annotation and example-construction pipeline as the original data: session boundaries provide coarse-grained supervision, GPT-4.1-mini annotates intra-session topic boundaries, and the annotated histories are converted into binary boundary-decision examples.

As shown in Table 8, reducing the training size to 50% lowers F1 from 0.483 to 0.399. In contrast, F1 remains within a relatively narrow range from 80% to 120% of the original size. Increasing the training size beyond the original setting yields little additional benefit: the 110% setting improves F1 by only 0.003 while increasing BOR, and the 120% setting has both lower F1 and a higher BOR than the original configuration. These results suggest diminishing returns from additional annotated dialogues within the tested range and support the adequacy of the original training-data setting. Although some smaller settings achieve a BOR closer to 1, they also have lower F1, highlighting the need to consider boundary accuracy and segmentation granularity jointly.

Effect of class distribution. We further examine the influence of the `CONTINUE : SEGMENT` ratio while keeping the total number of training examples identical across all settings. The original training data have an approximately $4:1$ ratio, reflecting the greater frequency of topic continuation in the annotated interaction streams. To construct equal-sized training sets with different class ratios, we supplement the candidate training pool with examples derived from additional non-abstention LongMemEval dialogue histories, using the same boundary-annotation and example-construction pipeline. All training hyperparameters remain fixed across these settings.

Table 9 shows that all tested alternatives to the original ratio produce more boundaries relative to the reference. Equalizing the classes produces a BOR of 2.228, indicating more than twice as many predicted boundaries as reference boundaries. Less aggressive changes also increase over-segmentation. In particular, the $3:1$ setting achieves nearly the same F1 as the original configuration but produces substantially more boundaries.

Among the tested class ratios, the original $4:1$ distribution achieves the BOR closest to 1 while maintaining competitive F1. Under the same training-data size and fixed training hyperparameters, these results support retaining the observed class distribution to balance boundary detection accuracy with the preservation of coherent evidence segments.

Effect of supervision mixture. We assess sensitivity to the composition of hard and soft boundary supervision. Hard labels correspond to explicit session boundaries, while LLM-annotated soft labels capture finer-grained intra-session topic shifts. In the original training set, the hard-to-soft label ratio is $0.902:1$, approximately $1:1$. Here, Hard:Soft refers to the relative contributions of the two supervision sources, rather than the `CONTINUE:SEGMENT` class ratio. All training hyperparameters remain fixed across the compared settings.

Table 9: Effect of the training class ratio on segmentation quality. C and S denote `CONTINUE` and `SEGMENT`, respectively. The original configuration is shown in bold.

Class ratio (C:S)	F1↑	BOR→ 1
1 : 1	0.421	2.228
2 : 1	0.477	1.276
3 : 1	0.484	1.511
4 : 1 **(original)**	**0.483**	**1.169**

Table 10: Effect of the hard/soft supervision mixture on segmentation quality under fixed training hyperparameters. The original hard-to-soft ratio is $0.902 : 1$. The original configuration is shown in bold.

Supervision setting	F1↑	BOR→ 1
Hard-only	0.226	0.505
Hard:Soft $= 2 : 1$	0.321	0.674
Hard:Soft $= 1 : 2$	0.459	0.816
Soft-only	0.472	1.907
Original mix ($0.902 : 1$)	**0.483**	**1.169**

As shown in Table 10, hard-only supervision produces conservative boundary predictions, resulting in low F1 and substantial under-segmentation. Incorporating soft labels improves boundary detection, but soft-only supervision leads to pronounced over-segmentation. The original mixture achieves both the highest F1 and the BOR closest to 1 among the tested configurations. These results support the complementary roles of the two supervision sources: explicit session boundaries provide coarse structural anchors, while LLM annotations supply finer-grained topic-transition signals. Their combination provides a favorable balance between boundary detection accuracy and segmentation granularity.

Together, these analyses support the adequacy of the original training-data setting and the complementary value of the two supervision sources. Performance remains relatively stable under moderate changes in training data size, whereas changes in supervision mixture or class distribution can substantially affect segmentation granularity. With training hyperparameters held fixed, the original configuration provides a practical balance between boundary detection accuracy and boundary-count calibration.

F.3 LLM-As-A-Judge Details

While LLM-as-a-Judge has become a common practice for long-term dialogue memory evaluation, we observe that a simple binary grading scheme can be overly permissive in this setting. When the judge only decides whether an answer is "correct" or "incorrect", vague, under-specified, or weakly supported predictions may be accepted as correct as long as they partially overlap with the gold answer. This is especially problematic for memory QA, where the key challenge is to recover the specific user-related fact, temporal relation, or preference evolution required by the question.

To obtain a more reliable evaluation, we use GPT-4.1-mini as the judge model and adopt a stricter four-way protocol: *correct*, *partially_correct*, *incorrect*, and *insufficient*. The *correct* label is used only when the prediction clearly matches the core fact in the gold answer, allowing for paraphrases, equivalent date formats, and non-conflicting extra details. The *partially_correct* label is used when the prediction contains relevant but incomplete information, while *incorrect* is used for wrong, contradictory, unrelated, or conflicting predictions. The *insufficient* label is used for answers that are too vague, evasive, or unverifiable. Only predictions labeled as *correct* are counted as correct, reducing false positives from incomplete or ambiguous answers.

Judge Prompt

```
You are an expert grader for memory QA.
Your job is to compare a model prediction with a gold answer and
    decide whether they match on the core fact.

Grading policy (important):
1) Be reasonably generous, not strict string-match.
2) If prediction clearly expresses the same core fact/topic as the
    gold answer, mark it correct even if wording differs.
3) Prediction may be longer than gold; extra non-conflicting detail
    is allowed.
4) For time answers, treat equivalent formats/references as correct (
    e.g., "May 7" vs "7 May"; relative wording that maps to same date/
    period).
5) Mark incorrect only when the core fact is wrong, missing,
    contradictory, or mostly unrelated.
6) Use partially_correct only when prediction contains a relevant but
     incomplete core answer.
7) Penalize contradictions to gold; if prediction includes both
    correct and incorrect conflicting claims, mark incorrect.

Return JSON only with keys:
- is_correct: boolean
- verdict: one of ["correct", "partially_correct", "incorrect", "
    insufficient"]
- reason: short explanation

Calibration:
- For verdict "correct", set is_correct=true.
- For verdict "partially_correct", set is_correct=false.
- For verdict "incorrect" or "insufficient", set is_correct=false.
```

F.4 Empirical Results on Qwen-family Models

To examine whether the observed improvements depend on a specific model family, we further conduct a cross-backbone evaluation using Qwen-family models. In the main experiments, we use GPT-4.1-mini as the general memory construction model and GPT-5-mini as the response model. In this appendix setting, we replace them with Qwen3.5-Flash and Qwen3.5-Plus, respectively. This setting changes both the memory-writing model and the response generator, and therefore should be interpreted as a cross-backbone robustness test rather than a component-wise ablation.

The two Qwen models play different roles in our pipeline. Qwen3.5-Flash is used as the general memory construction model, corresponding to GPT-4.1-mini in the main setting. It is designed as a faster and more cost-efficient model, which makes it suitable for large-scale memory construction but may be less stable for fine-grained memory extraction, summarization, and update operations. Qwen3.5-Plus is used as the response model, corresponding to GPT-5-mini in the main setting. Compared with Qwen3.5-Flash, it is a stronger model in the Qwen3.5 family and supports thinking-enabled response generation. This makes it particularly suitable for reasoning-oriented personalization tasks, but also increases query-time latency. Therefore, this appendix setting provides a useful stress test for both construction-time robustness and response-time reasoning ability.

Table 11 and Table 12 report the cross-backbone results with Qwen-family models. Overall, `Threader` remains the strongest overall method on both datasets, demonstrating that the advantage of structure-aware memory access is not tied to a specific model family. On LongMemEval, `Threader` achieves an overall accuracy of 87.00, outperforming the strongest baseline, EverMemOS, by 8.21% relative improvement. On PersonaMem, `Threader` also obtains the best overall score of 82.00, outperforming the strongest baseline, A-Mem, by 2.98% relative improvement.

On LongMemEval, the gains remain most pronounced on evidence-intensive question types. `Threader` achieves the best results on *Knowledge-Update* and *Temporal-Reasoning*, with scores

Table 11: **Main results on LongMemEval with Qwen-family models.** `Threader` retrieves the top-12 segments for LongMemEval. SS-User, SS-Assi., SS-Pref., Multi-S., Know.-U., and Temp.-R. denote Single-Session-User, Single-Session-Assistant, Single-Session-Preference, Multi-Session, Knowledge-Update, and Temporal-Reasoning, respectively. Results are grouped by backbone. Within each backbone block, the best result in each column is shown in **bold**, the second-best distinct result is underlined.

Method	SS-User	SS-Assi.	SS-Pref.	Multi-S.	Know.-U.	Temp-R.	Overall
			Qwen3.5-plus				
Self-RAG	90.00	94.64	60.00	69.17	79.49	80.45	79.00
HippoRAG2	77.14	96.43	80.00	72.93	88.46	71.43	78.60
Mem0	72.86	21.43	26.67	40.60	48.72	56.39	47.60
MemU	**98.57**	82.14	40.00	**82.71**	87.18	45.86	73.20
A-Mem	92.86	**98.21**	26.67	54.89	78.21	64.66	69.60
MemOS	81.43	89.29	**90.00**	60.15	83.33	47.37	68.40
EverMemOS	94.29	71.43	66.67	75.94	83.33	82.71	80.40
Threader	97.14 (↓1.45%)	**98.21** (↑0.00%)	70.00 (↓22.22%)	75.94 (↓8.18%)	**93.59** (↑5.80%)	**87.97** (↑6.36%)	**87.00** (↑8.21%)

Table 12: **Main results on PersonaMem with Qwen-family models.** `Threader` retrieves the top-8 segments for PersonaMem. Generalize-New-Scn., Preference-Align, User-Shared, User-Mentioned, Update-Reason, New-Ideas, and Preference-Evolution denote generalizing-to-new-scenarios, provide-preference-aligned-recommendations, recalling-user-shared-facts, recalling-facts-mentioned-by-the-user, recalling-reasons-behind-previous-updates, suggesting-new-ideas, and tracking-full-preference-evolution, respectively. Results are grouped by backbone. Within each backbone block, the best result in each column is shown in **bold**, and the second-best distinct result is underlined.

Method	Generalize -New-Scn.	Preference -Align	User-Shared	User-Mentioned	Update-Reason	New-Ideas	Preference Evolution	Overall
				Qwen3.5-plus				
Self-RAG	29.82	63.64	10.85	52.94	77.78	**59.14**	72.66	52.29
HippoRAG2	78.95	80.00	79.84	58.82	83.84	46.24	74.10	73.17
Mem0	64.91	54.55	37.21	58.82	83.84	39.78	86.33	61.97
MemU	61.40	72.73	81.39	64.71	85.86	38.71	76.26	70.97
A-Mem	91.23	**90.91**	73.64	**82.35**	**95.96**	50.54	83.45	79.63
MemOS	78.95	81.82	75.19	70.59	85.86	37.63	76.98	72.33
EverMemOS	91.23	83.64	**82.17**	64.71	91.92	35.48	84.89	77.59
Threader	**94.74** (↑3.85%)	**90.91** (→0.00%)	**82.17** (→0.00%)	76.47 (↓7.14%)	89.90 (↓6.32%)	52.69 (↓10.91%)	**87.77** (↑1.67%)	**82.00** (↑2.98%)

of 93.59 and 87.97, respectively. These categories require locating fine-grained historical evidence and preserving update or temporal relations, where lossy memory construction can be sensitive to the quality of the memory-writing model. This sensitivity is reflected in the performance of rewrite-heavy memory systems: Mem0 drops from 57.20 in the GPT-based setting to 47.60 in the Qwen-based setting, and EverMemOS drops from 84.80 to 80.40. Since these methods rely on the construction model to extract, summarize, merge, and update memory entries, replacing GPT-4.1-mini with Qwen3.5-Flash can affect the quality of the generated memory bank. In contrast, `Threader` avoids construction-time LLM rewriting and preserves raw dialogue evidence through topic-coherent segmentation, making its memory construction process less dependent on the summarization ability of a particular backbone. Although `Threader` also slightly decreases from 90.60 to 87.00, it preserves its overall advantage and remains strongest on the most evidence-intensive categories.

PersonaMem shows a different pattern. Under the Qwen backbone, multiple methods achieve substantially higher overall accuracy than in the GPT-based setting. For example, HippoRAG2 improves from 59.42 to 73.17, EverMemOS from 65.20 to 77.59, A-Mem from 60.78 to 79.63, and `Threader` from 69.95 to 82.00. These gains suggest that backbone choice plays an important role in PersonaMem performance. Many PersonaMem questions require reasoning about user preferences, tracking their evolution, and generalizing them to new scenarios. The improvements may therefore partly reflect Qwen3.5-Plus's ability to interpret and integrate retrieved evidence for these reasoning tasks.

Under the Qwen backbone, `Threader` achieves the best results on *generalizing-to-new-scenarios*, *provide-preference-aligned-recommendations*, *recall-user-shared-facts*, and *tracking-full-preference-*

Table 13: **Full-pipeline ablation of multi-view representation on LongMemEval**, with K_1 reduced from 40 to 10. Both variants retain BM25 retrieval, message-level dense scoring, and cross-encoder reranking.

Method	Accuracy (%)
THREADER w/o multi-view representation	78.4
THREADER (full pipeline)	**81.2**

evolution. This further shows that topic-coherent raw evidence retrieval can be effectively exploited by a different model family, especially when the response model has stronger reasoning ability. However, `Threader` is less advantageous on *recalling-the-reasons-behind-previous-updates*, *suggest-new-ideas*, and questions, where explicit abstraction of preference changes or higher-level user profiles can be beneficial. This observation is consistent with the trade-off observed in the main experiments: `Threader` prioritizes faithful evidence preservation and retrieval, while proactive preference abstraction remains complementary.

Overall, the Qwen-based results indicate that `Threader` is robust across model families and particularly effective when memory access depends on preserving distributed evidence. At the same time, they highlight an accuracy–efficiency trade-off: thinking-enabled response generation improves reasoning-heavy personalization performance, but substantially increases online serving latency.

F.5 ADDITIONAL ABLATION STUDY

F.5.1 ABLATION STUDY OF MULTI-VIEW REPRESENTATION IN THE FULL PIPELINE.

The dense-only ablation isolates the contribution of multi-view segment representations. We further examine whether this contribution remains beneficial when BM25, message-level dense scoring, and cross-encoder reranking are enabled. Multi-view representation primarily affects Stage I candidate retrieval, while Stage II refines the retrieved candidates. Its effect on final accuracy therefore depends on the candidate pool size $|\mathcal{C}|$. In the main setting, the max candidate pool size is 80 ($2 \times K_1$), which may allow relevant segments to enter Stage II both with and without multi-view representation, despite differences in their initial ranks. Subsequent refinement can then recover these segments in either setting, making final accuracy less sensitive to the quality of Stage I ranking.

To make the contribution of candidate selection more apparent, we reduce K_1 from 40 to 10 and compare the complete pipeline with a variant that removes multi-view representation at Top-5. Both variants retain BM25 retrieval, message-level dense scoring, and cross-encoder reranking. As shown in Table 13, multi-view representation improves accuracy from 78.4% to 81.2%, a gain of 2.8 percentage points. This result demonstrates its complementary value within the full retrieval pipeline under a more restrictive candidate budget: improving candidate selection remains important because downstream refinement can only operate on evidence admitted to Stage II.

F.5.2 ABLATION STUDY ON PERSONAMEM

Table 14 reports ablation results on PersonaMem along the same four design axes as the main experiments: segmentation, multi-view representation, retrieval signals, and retrieval granularity.

Segmentation ablation and cross-dataset transfer. Replacing the learned topic-coherent segmentation with fixed-size chunking consistently reduces answer accuracy. THREADER achieves 69.95%, outperforming the strongest fixed-size alternative, Chunk-500, by 2.55 percentage points. Notably, the segmenter is trained exclusively on LongMemEval dialogue histories and *directly transferred to PersonaMem*, with the same model parameters and decision threshold, without dataset-specific fine-tuning or threshold adjustment. Despite the substantial differences in session length and dialogue structure between the two benchmarks, the transferred segmenter remains effective for downstream memory retrieval. These results support the cross-dataset generalization of the segmentation approach and show that its benefits extend beyond the benchmark used for training.

Table 14: **Ablation study on the key design axes of `Threader` on Personamem.** We evaluate the impact of segmentation, multi-view representation, retrieval signals, and retrieval granularity on answer accuracy. Emb.-User, Emb.-Asst., and Emb.-Full correspond to using only user, assistant, or full interaction views, respectively.

Variant	ACC
Segmentation ablation	
`Threader`	69.95
Chunk-300.	65.70
Chunk-500	67.40
Chunk-700	63.67
Chunk-1K	59.76
Multi-view representation ablation	
Emb. w/o Msg.	58.40
Emb.-Full	58.23
Emb.-User	58.23
Emb.-Asst.	54.67

Variant	ACC
Retrieval signal ablation	
`Threader`	69.95
w/o Emb.	67.40
w/o BM25	65.87
w/o Rerank.	66.04
w/o Emb.&BM25	62.82
w/o Emb.&Rerank.	66.55
w/o BM25&Rerank.	62.14
w/o Msg.-Emb.	65.70
Retrieval granularity ablation	
Segment Unit	69.95
Message Unit	67.06

Representation and retrieval ablations. The remaining ablations broadly align with the findings on LongMemEval. Combining segment views performs competitively with or better than individual views, although the gains over user-only and full-interaction representations are modest. Removing retrieval signals or message-level refinement degrades performance, supporting the value of combining complementary relevance signals and localized evidence matching. Retrieving individual messages instead of complete segments also reduces accuracy, highlighting the importance of preserving contextual coherence. Together, these results support the effectiveness of THREADER's representation and retrieval design across the two benchmarks.

F.6 RETRIEVAL HYPERPARAMETER SENSITIVITY

We examine how changes in the retrieval weights affect evidence recall in `Threader`. We calibrate the segmentation threshold and retrieval weights on the same 50-instance LongMemEval subset used for segmenter evaluation in Appendix F.2, using segmentation quality and evidence recall, respectively. We then freeze the trained segmenter and these calibrated settings and apply them unchanged to PersonaMem, without further fine-tuning or calibration. Our main configuration uses $(w_u, w_a, w_f) = (0.5, 0.25, 0.25)$, $\beta = 0.65$, and $\alpha = 0.3$. The analyses below characterize sensitivity to variations in these weights.

The view weights (w_u, w_a, w_f) combine query–segment cosine similarities for the user-only, assistant-only, and full-interaction representations in Eq. 3. The parameter β controls the contribution of BM25 relative to the refined message-level dense score in Eq. 6. The parameter α balances this hybrid score with the segment-level cross-encoder reranking score in Eq. 7. Unless otherwise specified, we fix the coarse-retrieval budget to $K_1 = 40$, and the message-level scoring budget to $K_2 = 2$.

Segment-view Weight Sensitivity. We isolate the effect of segment-view weights using segment-level dense retrieval alone, with BM25 matching and message-level dense matching disabled. Table 15 shows that increasing the user-only view weight does not monotonically improve evidence recall. The relative performance of the weighting schemes also depends on the retrieval budget: $(0.7, 0.15, 0.15)$ performs best at Top-10, whereas the main configuration $(0.5, 0.25, 0.25)$ performs best at Top-5 and Top-12 and remains competitive at Top-8. Across the five evaluated configurations, Top-12 Full-R ranges from 84.89% to 86.81%, and Mean-R from 90.41% to 91.65%. These results show a budget-dependent trade-off between emphasizing user-side evidence and retaining complementary assistant-side and full-interaction signals, with relatively modest variation at Top-12.

Table 15: **Segment-view weight analysis.** We evaluate different dense segment-view weights (w_u, w_a, w_f) using segment-level dense retrieval only, with BM25 matching and message-level dense matching disabled. Full-R denotes the fraction of queries with complete evidence coverage, and Mean-R denotes average evidence recall.

(w_u, w_a, w_f)	Top-5		Top-8		Top-10		Top-12	
	Full-R	Mean-R	Full-R	Mean-R	Full-R	Mean-R	Full-R	Mean-R
$(0.3, 0.35, 0.35)$	0.6596	0.7644	0.7638	0.8464	0.8064	0.8758	0.8489	0.9041
$(0.4, 0.3, 0.3)$	0.6660	0.7666	**0.7723**	**0.8541**	0.8128	0.8807	0.8617	0.9129
$(0.5, 0.25, 0.25)$	**0.6745**	**0.7734**	**0.7723**	0.8534	0.8170	0.8817	**0.8681**	**0.9165**
$(0.6, 0.2, 0.2)$	0.6553	0.7625	**0.7723**	0.8534	0.8213	0.8832	0.8638	0.9141
$(0.7, 0.15, 0.15)$	0.6511	0.7509	0.7702	0.8524	**0.8277**	**0.8856**	0.8596	0.9141

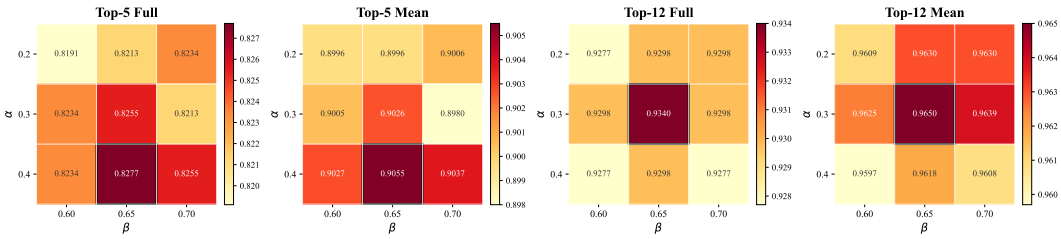


Figure 7: **Fusion weight sensitivity analysis.** Performance heatmaps of different BM25 interpolation weights β and final fusion weights α under Top-5 and Top-12 retrieval settings.

Fusion Weight Sensitivity. Holding the segment-view weights at $(0.5, 0.25, 0.25)$, we vary $\beta \in \{0.60, 0.65, 0.70\}$ and $\alpha \in \{0.2, 0.3, 0.4\}$. Figure 7 reports evidence recall across this local grid. At Top-12, Full-R ranges from 92.77% to 93.40%, and Mean-R from 95.97% to 96.50%, corresponding to spans of 0.63 and 0.53 percentage points. At Top-5, the corresponding spans are 0.86 and 0.75 percentage points. Thus, both recall metrics vary by less than one percentage point across the nine displayed configurations at each retrieval budget, indicating limited sensitivity to small fusion-weight changes within the examined range.

F.7 Abstraction and Preference-Based Tasks

Preserving conversational evidence and constructing abstract memories involve different design trade-offs. `Threader` retains original interactions and defers higher-level synthesis to response generation, whereas rewrite-based methods perform part of this abstraction during memory construction. Consequently, tasks involving preference inference, preference evolution, or personalized idea generation may depend strongly on the response model's ability to synthesize retrieved evidence. We investigate this issue through evidence inspection and a controlled comparison of response models.

Diagnosing retrieval and generation errors. We investigate whether `Threader`'s limitations on preference-related tasks arise from missing evidence during retrieval or from insufficient synthesis during response generation. We manually inspect the evidence retrieved for LongMemEval single-session preference questions and observe 100% evidence recall. Nevertheless, answer accuracy with GPT-5-mini is 96.67%. Thus, the remaining errors on this task occur despite the availability of the required evidence, identifying response generation as the bottleneck.

Improving generation with the same retrieved evidence. To further examine this distinction, we keep each method's memory bank, retrieved evidence, and response prompt unchanged, and replace GPT-5-mini with GPT-5.6-sol using maximum reasoning effort. As shown in Table 16, `Threader`'s accuracy across the three selected tasks increases from 60.69% to 75.95%, achieving the highest overall score among the compared methods. Its single-session preference accuracy reaches 100%, and it also achieves the best performance on New-Ideas. Because these improvements are obtained from the same retrieved evidence, they demonstrate that the preserved memory units contain information that a more capable response model can use more effectively.

Table 16: Accuracy (%) on three selected abstraction-related question types. SS-Pref. is from LongMemEval; New-Ideas and Preference Evolution are from PersonaMem. Overall denotes accuracy pooled across the 262 questions in these three categories. GPT-5.6-sol uses maximum reasoning effort. The best result in each metric is shown in bold.

Memory method	Response model	SS-Pref.	New-Ideas	Preference Evolution	Overall
Mem0	GPT-5-mini	26.67	27.96	66.91	48.47
Mem0	GPT-5.6-sol	46.67	48.39	58.99	53.82
EverMemOS	GPT-5-mini	96.67	26.88	64.03	54.58
EverMemOS	GPT-5.6-sol	**100.00**	59.14	**79.14**	74.43
Threader	GPT-5-mini	96.67	45.16	63.31	60.69
Threader	GPT-5.6-sol	**100.00**	**65.59**	77.70	**75.95**

The importance of evidence preservation. These results highlight the distinction between retrieving sufficient evidence and successfully synthesizing it into an answer. The evidence audit directly identifies a generation bottleneck on single-session preference questions, while the controlled response-model comparison demonstrates the substantial role of generation across the broader set of preference-related tasks. Crucially, preserving the original evidence allows stronger downstream reasoning to improve answers without reconstructing the memory bank or retrieving different content. This supports `Threader`'s design principle: retaining fine-grained evidence provides a reliable basis for query-time abstraction. Combining this evidence-preserving foundation with lightweight summaries remains a complementary direction, as discussed in AppendixA.1.

G RESPONSE PROMPT

Response Prompt for LongMemEval

```
You are a response agent. You will receive:
1) a user question
2) a retrieved History Session (time-ordered, possibly multi-session).


Answer using ONLY the History Session. Do not invent facts.

Question: {question}

History Session:
{history_session}

Rules:
1) Use only supported evidence from History Session; combine evidence
     across segments/sessions when needed.
2) If facts conflict over time, use the most recent supported fact
    unless the question explicitly asks for history/comparison.
3) Keep names, numbers, dates, places, and units exact.
4) For count/sum/difference/time-window questions, first apply the
    temporal condition (before/after/within-last-X), then compute.
5) If Question contains "[Question Date: ...]", use it as the anchor
    for relative time expressions; otherwise use the best explicit
    reference time in History Session.
6) If evidence is sufficient, answer directly. If insufficient for a
    required value, state the missing key fact (do not speculate).
7) For preference-related questions (including suggestion/tips),
    prioritize user-specific established preferences and constraints
    from History Session; avoid generic advice that is not grounded
    in history.
```

```
8) Do not claim "not mentioned/no history" when relevant evidence
    exists in History Session.
9) Output exactly one final answer target. Do not provide multiple
    candidates, unnecessary alternatives, or contrastive add-ons.
10) If ordering is requested, return an ordered list (1., 2., 3.,
    ...); otherwise keep a single concise answer sentence.
11) For preference questions, use:
Preferred: <core preference>. Avoid/Constraints: <constraints/non-
    preferences>.

Output:
- Output ONLY the final answer (no analysis, no tags).
```

Response Prompt for PersonaMem

```
You are solving a PersonaMem multiple-choice memory task.

Inputs:
1) Retrieved conversation history (memory evidence)
2) Current user question
3) Four answer options

Task:
Pick the ONE option (A/B/C/D) best supported by user-specific memory.

Internal Decision Rules:
1) Build a compact memory profile from history: likes/dislikes,
    constraints, recent updates, and reasons for changes (earlier ->
    later).
2) Score each option by:
   a) Memory fit: specific match to user facts/preferences/
    constraints.
   b) Temporal fit: respects latest state and correct change
    direction when evolution is asked.
   c) Transfer quality (for recommendation/generalizing): applies the
     user's stable patterns (social intensity, effort tolerance, pace,
     style, context) to the new scenario.
   d) Conflict penalty: contradicts explicit dislikes, failed
    experiences, or later updates.
   e) Unsupported-detail penalty: adds specific entities/claims not
    evidenced in history.
3) Tie-break: prefer the option with stronger personalized evidence
    and fewer unsupported assumptions; do not prefer generic but
    plausible advice.
4) For recall-style questions, exact factual consistency is primary;
    for recommendation/generalizing questions, preference-and-
    constraint transfer is primary.

Hard Constraints:
- Use only retrieved history; no outside knowledge guessing.
- This is option selection, not open-ended answering.
- Output one letter only.

Output Format (strict):
- A single uppercase letter: A, B, C, or D
- Nothing else

[Retrieved History]
{history_session}

[Question]
{question}
```

```
[Options]
{answer_options}
```