EverMemOS: A Self-Organizing Memory Operating System for Structured Long-Horizon Reasoning

Chuanrui Hu[1,2*], **Xingze Gao**[1,2*], **Zuyi Zhou**[1,2*], **Dannong Xu**[1,2], **Yi Bai**[1,2], **Xintong Li**[1,2], **Hui Zhang**[1,2], **Tong Li**[1,2], **Chong Zhang**[2], **Lidong Bing**[2†], **Yafeng Deng**[1,2†]

[1]EverMind [2]Shanda Group

{chuanrui.hu, xingze.gao, zuyi.zhou, dannong.xu, baiyi, xintong.li, zhanghui, litong02, zhangchong, lidong.bing, dengyafeng}@shanda.com

Abstract

Large Language Models (LLMs) are increasingly deployed as long-term interactive agents, yet their limited context windows make it difficult to sustain coherent behavior over extended interactions. Existing memory systems for LLMs often store isolated records and retrieve fragments, limiting their ability to consolidate evolving experience and resolve conflicts. We introduce **EverMemOS**, a self-organizing memory operating system that implements an engram-inspired lifecycle for computational memory. First, *Episodic Trace Formation* converts dialogue streams into *MemCells* that capture episodic traces, atomic facts, and time-bounded foresight. Second, *Semantic Consolidation* organizes MemCells into thematic *MemScenes*, distilling stable semantic structures and updating user profiles. Finally, *Reconstructive Recollection* performs *MemScene*-guided agentic retrieval to compose the necessary and sufficient context for downstream reasoning. Experiments on LoCoMo, LongMemEval, and PersonaMem-v2 show that EverMemOS significantly outperforms state-of-the-art methods on memory-augmented reasoning tasks. Our code is available at https://github.com/EverMind-AI/EverOS.

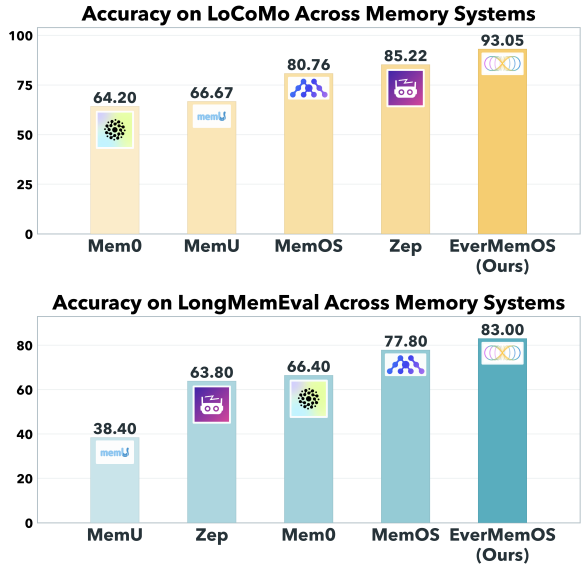

Figure 1: Evaluation results of different memory methods for LLMs on two benchmarks (LoCoMo and LongMemEval). All methods are based on GPT-4.1-mini.

1 Introduction

Large Language Models (LLMs) are increasingly deployed as long-term interactive agents rather than transient conversational tools (Yehudai et al., 2025; Ferrag et al., 2025). For providing better personalized services, LLM-based agents must maintain consistent personas and user models over extended interactions while continuously incorporating new constraints over extended timeframes, spanning days, months, or even years. To address this challenge, expanding context windows is a direct approach, but ultra-long contexts still degrade in performance (e.g., the "Lost-in-the-Middle" phenomenon) and incur prohibitive computational costs (Liu et al., 2024). Consequently, recent research has increasingly focused on constructing memory for LLMs that can both store past information and organize experiences into coherent, evolving structures that support long-horizon reasoning (Wu et al., 2025; Maharana et al., 2024).

Recently, a broad range of memory-augmented approaches have been proposed, including retrieval-based memory (Zhong et al., 2024; Packer et al., 2024), trainable memory (Zheng et al., 2024; Gong et al., 2024), and more recently *Memory Operating Systems* that unify storage, retrieval, filtering, and updating (Li et al., 2025; Kang et al., 2025). However, enabling long-term consistency in reasoning remains challenging. While these methods improve scalability and modularity, most of them treat memory as flat collections of isolated records. As a re-

[*] Equal contribution.
[†] Corresponding author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 45836–45853
July 2-7, 2026

sult, many failures stem not from missing information but from poor integration, where fragmented experiences are not consolidated into higher-level semantic structures. Without consolidation and abstraction, agents may retrieve relevant facts yet fail to detect conflicts, maintain stable user models, or reason consistently over time. **Therefore, a key limitation of existing memory methods is the absence of an explicit mechanism to transform fragmented episodic experiences into coherent and stable knowledge structures that support long-horizon reasoning.**

To address the above limitation, we propose **EverMemOS**, a unified and deployment-oriented Memory Operating System that models memory as a dynamic lifecycle for long-term LLM-based agents. As shown in Figure 1, EverMemOS significantly outperforms the state-of-the-art memory methods for LLMs in experimental evaluation, relatively improving overall accuracy by 9.2% on LoCoMo and 6.7% on LongMemEval compared to the strongest baseline method. EverMemOS aims to transform fragmented episodic experiences into coherent and stable knowledge structures that support long-horizon reasoning through three phases. First, **Episodic Trace Formation** transforms the unbounded stream of interaction history into discrete, stable memory traces (termed *MemCells*). Second, **Semantic Consolidation** transforms MemCells into stable, scene-level structures (termed *MemScenes*) that support coherent aggregation, such as maintaining consistent user profiles across interactions. Finally, **Reconstructive Recollection**, guided by the goal of *necessity and sufficiency*, actively composes only the grounded context required for a given query and supports long-horizon reasoning, rather than indiscriminately retrieving all potentially relevant records.

EverMemOS does not aim to simulate biological memory at the neural level. Instead, it draws on organizing principles from biological memory systems and translates them into a computational framework. Figure 2 illustrates the intuition behind EverMemOS. A fragment-based system may recall a user's preference for IPA and recommend an alcoholic drink, failing to account for a newly introduced constraint that the user is taking antibiotics. In contrast, EverMemOS consolidates these experiences into a coherent representation of the user's state, enabling the agent to safely recommend a non-alcoholic alternative. Although such foresight-oriented behaviors are not explicitly captured by existing benchmarks, they expose a fundamental limitation of fragment-based memory and motivate the system-level design of EverMemOS. Empirically, comprehensive experiments on three benchmarks for memory-augmented reasoning consistently indicate the superiority of EverMemOS, compared to the state-of-the-art methods.

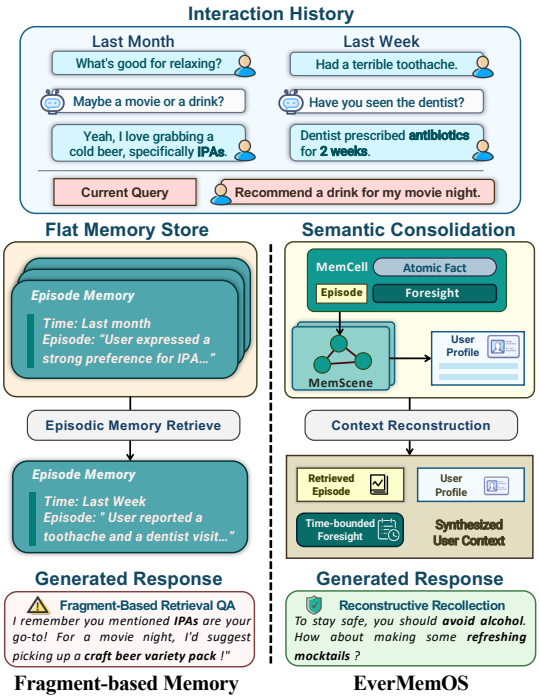

Figure 2: Comparison of typical fragment-based memory and EverMemOS in an interactive chat scenario.

Our contributions are summarized as follows:

- **System Design:** We introduce **EverMemOS**, a unified and deployment-oriented Memory Operating System for LLMs that reconceptualizes memory as a lifecycle, shifting from passive storage of records to structured organization of experience.
- **Innovative Method:** We propose a three-phase method that can transform fragmented episodic experiences into coherent and stable knowledge structures that support long-horizon reasoning.
- **Empirical Validation:** Experimental results demonstrate that EverMemOS achieves state-of-the-art performance on multiple long-context benchmarks for memory-augmented reasoning, validating the effectiveness of lifecycle-based memory organization.

2 Related Work

2.1 Memory Mechanisms in LLMs

Context Window Extension. Large language models (LLMs) are constrained by fixed-length context windows. Prior work extends context via sparse attention (Beltagy et al., 2020; Zaheer et al., 2020), recurrence (Dai et al., 2019; Bulatov et al., 2022), and length extrapolation (Chen et al., 2024, 2025). However, longer context does not guarantee effective utilization: the "Lost-in-the-Middle" phenomenon persists (Liu et al., 2024; Bulatov et al., 2023), suggesting context extension alone is insufficient for durable memory.

Retrieval-Augmented and Parametric Memory. Retrieval-augmented generation (RAG) (Lewis et al., 2020) externalizes memory to alleviate window limits, but its reliability depends on retrieval quality (Ram et al., 2023). Parametric approaches internalize information, yet often suffer from forgetting and instability (De Lange et al., 2022). Hybrid approaches (Wang et al., 2023; Packer et al., 2024) alleviate issues but lack a unified organizational principle for persistent memory.

2.2 Memory Systems

Early Computational Memory. Early differentiable memory systems (e.g., NTM/DNC/Key–Value memories) (Graves et al., 2014, 2016; Miller et al., 2016) introduced external memory interaction, but scale poorly and are ill-suited to modern autoregressive LLMs.

Memory in LLM Agents. As LLM-based agents evolve (Xi et al., 2023; Xia et al., 2024), memory systems have shifted toward persistent state integration. Recent systems introduce episodic (Wang and Chen, 2025), semantic (Shinn et al., 2024), and hierarchical task memory (Sun and Zeng, 2025). However, many designs still rely on fragmented text units and limited consolidation, which can degrade long-horizon performance (Packer et al., 2024).

Memory Operating Systems. Recent work formalizes memory management as a system-level runtime. Some focus on lifecycle and capacity, such as *Nemori*'s (Nan et al., 2025) prediction-driven updates and *MemoryOS*'s (Kang et al., 2025) hierarchical control. Others, like *Mem0* (Chhikara et al., 2025) and *Zep* (Rasmussen et al., 2025), prioritize structured fact maintenance via knowledge graphs, while *MemOS* (Li et al., 2025) targets unified scheduling across memory types.

While these systems advance structural organization, they primarily focus on *storage optimization* or *fact maintenance*. EverMemOS distinguishes itself by implementing a *three-phase memory lifecycle* that transforms episodic traces into synthesized semantic structures for long-horizon reasoning.

3 EverMemOS

3.1 Framework Overview

Drawing inspiration from the biological engram lifecycle (Josselyn et al., 2015), EverMemOS follows a three-phase workflow (Figure 3): (1) **Episodic Trace Formation** encodes interaction streams into *MemCells*; (2) **Semantic Consolidation** organizes MemCells into *MemScenes* and updates user profiles; and (3) **Reconstructive Recollection** performs *MemScene*-guided retrieval under the goal of necessity and sufficiency.

3.2 Memory Primitives

At the core of EverMemOS is the **MemCell**, the atomic unit bridging low-level data and high-level semantics. Formally, a MemCell c is a tuple $c = (E, \mathcal{F}, P, M)$, where:

- E (**Episode**): A concise third-person narrative of the event, serving as the semantic anchor.
- $\mathcal{F} = \{f_1, \ldots, f_n\}$ (**Atomic Facts**): Discrete, verifiable statements derived from E for high-precision matching.
- P (**Foresight**): Forward-looking inferences (prospections; e.g., plans and temporary states) annotated with validity intervals $[t_{start}, t_{end}]$ to support temporal awareness.
- M (**Metadata**): Contextual grounding including timestamps and source pointers.

This structure turns memory from a static record $(E, \mathcal{F})$ into a temporally grounded representation that also supports **Foresight** (P).

3.3 Phase I: Episodic Trace Formation

Grounded in the engram concept (Josselyn et al., 2015), this first phase transforms the unbounded stream of interaction history $\mathcal{D} = \{d_1, \ldots, d_T\}$ into discrete, stable memory traces (MemCells). This process adopts a three-step pipeline to distill semantic signal from noisy interaction data:

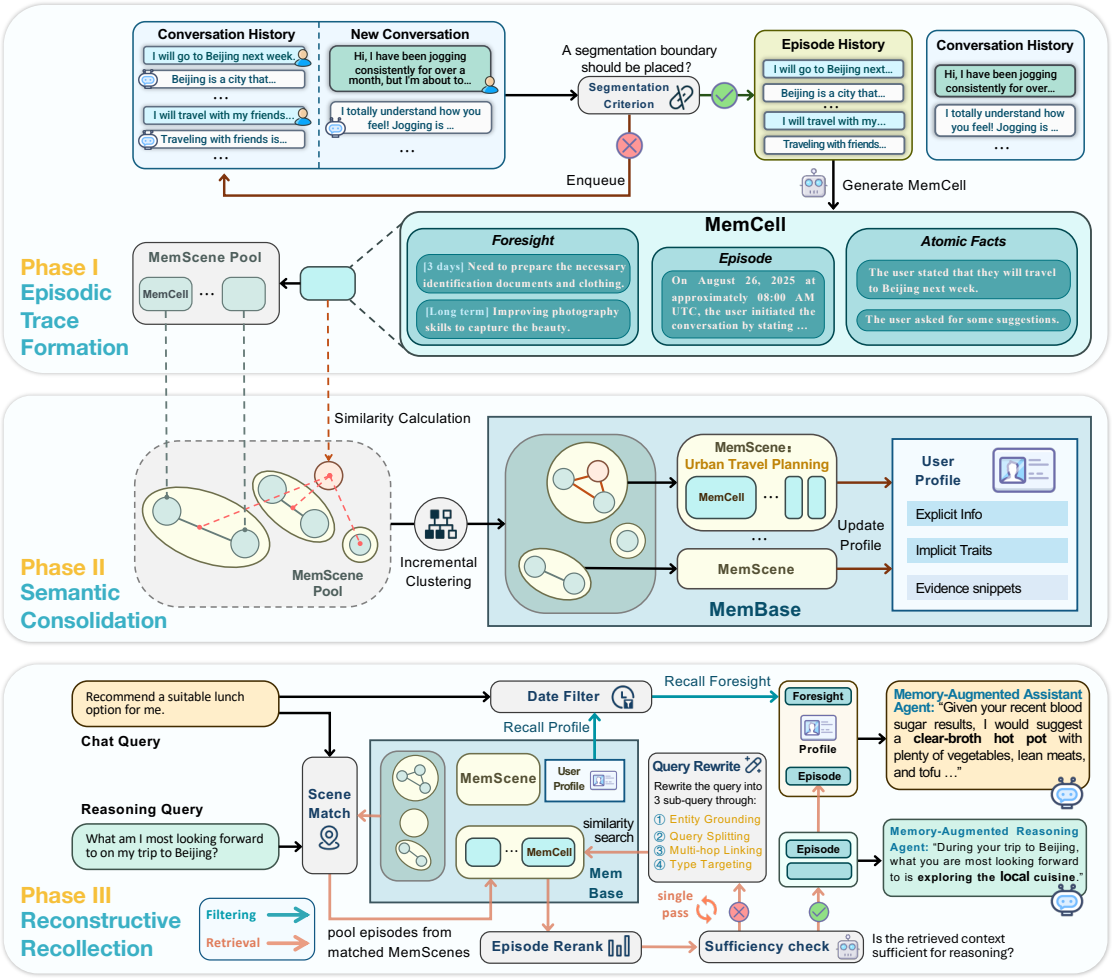

Figure 3: The EverMemOS workflow mirrors an engram-inspired memory lifecycle: (1) **Episodic Trace Formation** segments continuous dialogue into *MemCells* with episodes, atomic facts, and time-bounded foresight. (2) **Semantic Consolidation** organizes MemCells into *MemScenes* and updates a user profile. (3) **Reconstructive Recollection** performs *MemScene*-guided retrieval to compose the necessary and sufficient context.

Contextual Segmentation To discretize continuous streams, a **Semantic Boundary Detector** processes interactions via a sliding window. Upon detecting a topic shift, accumulated turns are encapsulated as a raw *episode history*. We implement this step via LLM prompting; while boundary detection is not perfect, we find it robust in downstream evaluation (see Table 3).

Narrative Synthesis To resolve dialogue redundancy and ambiguity, the episode history is synthesized into a high-fidelity **Episode** (E). This rewriting process produces a concise, third-person narrative with resolved coreferences, establishing a stable semantic anchor.

Structural Derivation From E, the system extracts **Atomic Facts** ($\mathcal{F}$) for precise matching and generates **Foresight** signals (P) with inferred validity intervals (e.g., distinguishing temporary "flu" from permanent "graduation"). Concretely, we prompt the LLM over the rewritten Episode E to output a constrained schema of Atomic Facts and Foresight signals with validity intervals $[t_{start}, t_{end}]$. These components are bundled with metadata M to form the final MemCell c.

3.4 Phase II: Semantic Consolidation

Inspired by systems consolidation (McGaugh, 2000), EverMemOS employs an online mechanism that organizes MemCells into higher-order structures to transition from transient episodes to stable long-term knowledge.

Incremental Semantic Clustering EverMemOS organizes memory dynamically. Formally, a **MemScene** is defined as $S = (\mathcal{C}, \mathbf{e}_{\text{centroid}}, t_{\text{last}})$. Here, $\mathcal{C}$ is the set of member MemCells, $\mathbf{e}_{\text{centroid}}$ is a running-mean embedding, and t_{last} tracks the most recent timestamp for temporal gating. When a new MemCell c arrives, the system computes its em-

bedding and retrieves the nearest MemScene centroid. A MemCell c is assimilated into S only when three conditions are jointly satisfied: (i) $\text{sim}(c, \mathbf{e}_{\text{centroid}}(S)) > \tau$, (ii) the temporal gap between c and the latest cell in S is within $\Delta_{\max}$, and (iii) no profile-level conflict is detected. If all conditions hold, c is merged and S is incrementally updated. Otherwise, a new MemScene is instantiated. This online process maintains thematic structure in real-time without batch reprocessing. The full algorithm is provided in Appendix B.1. EverMemOS uses a hybrid storage architecture to underpin this process. Raw MemCell text is persisted in a document store to preserve full fidelity, since embeddings are lossy and cannot reconstruct precise values such as numerical measurements or temporal intervals. Separate vector and keyword indices support dense and sparse retrieval respectively. This design decouples storage from indexing, and allows each component to scale horizontally.

Scene-Driven Profile Evolution Scene-level consolidation can also update a compact **User Profile** from aggregated evidence. When a new MemCell is assimilated into a **MemScene**, EverMemOS updates a concise scene summary and refreshes the user profile by prompting over these summaries (rather than individual turns), helping separate stable traits from temporary states. We maintain a compact profile of *explicit facts* (including time-varying measurements) and *implicit traits*, updated online from scene summaries with recency-aware updates and conflict tracking (Appendix B.5).

3.5 Phase III: Reconstructive Recollection

Building on theories of reconstructive memory (Schacter, 2008), retrieval in EverMemOS is modeled not as a static lookup but as an active **Reconstruction** process, guided by the goal of *necessity and sufficiency*. Given a query q, EverMemOS performs **agentic retrieval** grounded in *MemScenes*.

MemScene Selection We first compute relevance between the query and all MemCells by fusing dense and BM25 retrieval over their Atomic Facts $\mathcal{F}$ via Reciprocal Rank Fusion (RRF). We then score each MemScene by the maximum relevance among its constituent MemCells and select a small set of the highest-scoring MemScenes.

Episode and Foresight Filtering Within the selected MemScenes, we pool Episodes from their constituent MemCells and **re-rank** them to select a compact set for downstream inference. We then apply **Foresight Filtering**, retaining only time-valid Foresight whose validity intervals satisfy $t_{now} \in [t_{start}, t_{end}]$ (discarding expired ones).

Agentic Verification and Query Rewriting The retrieved context is evaluated by an LLM-based verifier for sufficiency. If it is deemed insufficient, the system triggers a **query rewriting** step to supplement retrieval; otherwise, the context is passed to the downstream module. Prompt templates are provided in Appendix C.1.

Task Modes We consider two downstream settings that share the same retrieval pipeline: **Memory-Augmented Reasoning** and **Memory-Augmented Chat**. For *Reasoning*, we use the retrieved Episodes as context for benchmark evaluation. For *Chat*, the composed context additionally incorporates the User Profile and time-valid Foresight signals, filtered by the current time $t_{now} \in [t_{start}, t_{end}]$; since these capabilities are not covered by existing reasoning benchmarks, we present them through qualitative case studies.

4 Experiments

We evaluate EverMemOS on two long-horizon memory-augmented reasoning benchmarks (LoCoMo (Maharana et al., 2024) and LongMemEval (Wu et al., 2025)), and report a profile study on PersonaMem-v2 (Jiang et al., 2025).

4.1 Experimental Setup

Benchmarks We evaluate memory-augmented reasoning on LoCoMo and LongMemEval. LoCoMo contains 1,540 questions over 10 ultra-long dialogues (∼9K tokens each), spanning single-hop, multi-hop, and temporal questions. LongMemEval (S-setting, ∼115k tokens per conversation) evaluates 500 questions requiring full-history parsing across core capabilities (e.g., updates and abstention). We additionally evaluate user profiling on PersonaMem-v2.

Baselines We compare EverMemOS against state-of-the-art memory systems: **Zep** (Rasmussen et al., 2025), **Mem0** (Chhikara et al., 2025), **MemOS** (Li et al., 2025), **MemoryOS** (Kang et al., 2025), and **MemU**[1]. **Fair comparison:** We standardize the answer-generation backbone

[1]Open-source memory infrastructure: https://github.com/NevaMind-AI/memU

across methods while keeping each baseline's official memory configuration unchanged; for LongMemEval, we report baseline scores from the official **MemOS** leaderboard. Full settings are provided in Appendix A.1.

Evaluation Protocol We adopt the **LLM-as-a-judge** protocol, following MemOS: each answer is evaluated by GPT-4o-mini and two auxiliary judge models, and scores are averaged across the three judgments in a *blind* setting. We validate the reliability of this protocol against human annotations in Section A.2 (Appendix), showing high agreement (Cohen's $\kappa > 0.89$).

Implementation Details EverMemOS uses GPT-4.1-mini (or GPT-4o-mini where specified) for all reasoning and memory operations. Retrieval uses hybrid dense+BM25 fusion (RRF) with reranking. Default retrieval hyperparameters are in Appendix A.1. Unless otherwise specified, quantitative experiments use **Memory-Augmented Reasoning**. We provide a token-level cost breakdown by lifecycle phase in Appendix (Table 8).

4.2 Main Results

Main results on two benchmarks are reported in Tables 1-2. We make three observations:

(1) **Lifecycle-driven performance gains.** EverMemOS outperforms the strongest baseline on each benchmark overall, i.e., Zep on LoCoMo by 7.0% and 9.2%, and MemOS on LongMemEval by 6.7%. We attribute this to the shift from flat memory storage to a structured lifecycle, which consolidates fragmented experiences into usable knowledge before retrieval, providing a more robust context than isolated record matching.

(2) **Structural consolidation aids complex reasoning that requires integrating dispersed evidence.** We can observe significant gains on LoCoMo multi-hop (+19.7%) and temporal (+10.0%) tasks, as well as LongMemEval knowledge update (+20.6%), validating the effectiveness of MemScenes. By clustering related episodes into coherent thematic units, EverMemOS presents the solver with a complete narrative context. This enables LLMs to naturally bridge dispersed evidence and resolve state conflicts that confuse other models relying on fragmented retrieval.

(3) **EverMemOS offers a favorable accuracy-efficiency trade-off.** As shown in Figure 6 , EverMemOS attains high accuracy with moderate retrieval budgets. This efficiency confirms the utility of the Reconstructive Recollection phase, where the agentic sufficiency check ensures the context is composed of necessary and sufficient evidence, avoiding the noise accumulation common in fixed-budget retrieval.

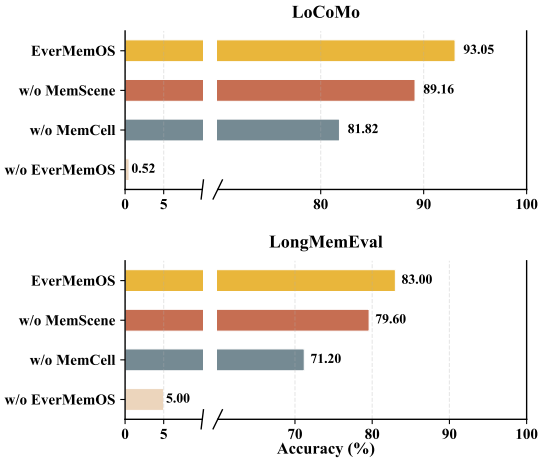

Figure 4: Ablation results (overall accuracy) on LoCoMo and LongMemEval.

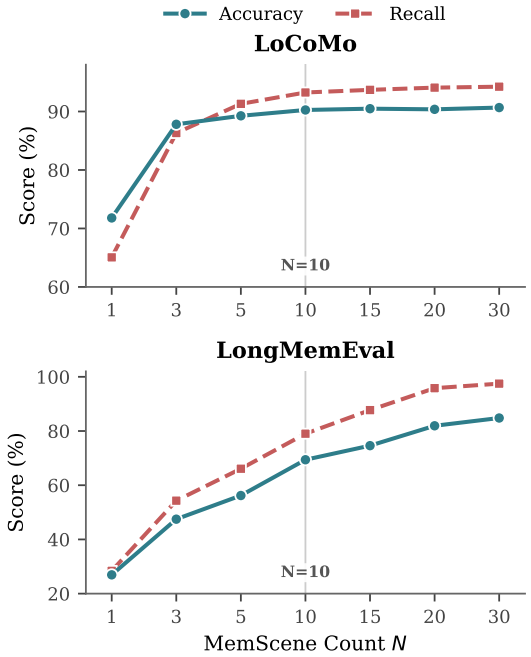

Figure 5: Sensitivity analysis on the MemScene count (N).

4.3 Ablation Study

We conduct ablations on LoCoMo to isolate the contributions of *MemScenes*, *MemCells*, and episode segmentation.

Impact of Memory Architecture. To isolate the contribution of memory structure, we compare EverMemOS with three degraded variants: w/o EverMemOS (no external memory), w/o MemScene (flat retrieval over MemCells), and w/o MemCell

Method	Avg. Tokens	Single Hop	Multi Hop	Temporal	Open Domain	Overall
GPT-4o-mini backbone						
MemoryOS	5.2k	62.43	56.50	37.18	40.28	54.70
Mem0	1.0k	66.71	58.16	55.45	40.62	61.00
MemU	4.0k	72.77	62.41	33.96	46.88	61.15
MemOS	2.5k	81.45	69.15	72.27	60.42	75.87
Zep	1.4k	88.11	71.99	74.45	66.67	81.06
EverMemOS	2.5k	**91.08** (↑3.4%)	**86.17** (↑19.7%)	**81.93** (↑10.0%)	**66.67** (↑0.0%)	**86.76** (↑7.0%)
GPT-4.1-mini backbone						
MemoryOS	5.5k	67.30	59.34	42.26	59.03	60.11
Mem0	1.0k	68.97	61.70	58.26	50.00	64.20
MemU	4.0k	74.91	72.34	43.61	54.17	66.67
MemOS	2.5k	85.37	79.43	75.08	64.58	80.76
Zep	1.4k	90.84	81.91	77.26	75.00	85.22
EverMemOS	2.3k	**96.67** (↑6.4%)	**91.84** (↑12.1%)	**89.72** (↑16.1%)	**76.04** (↑1.4%)	**93.05** (↑9.2%)

Table 1: Main results on **LoCoMo** under two backbones. All metrics are accuracy (%), except Avg. Tokens. For EverMemOS, values in parentheses denote relative change (%) compared to the strongest baseline under the same backbone.

Method	Token	SS-User	SS-Asst	SS-Pref	Multi-S	Know. Upd	Temp. Reas	Overall
MemU	0.5k	67.14	19.64	76.67	42.10	41.02	17.29	38.40
Zep	1.6k	92.90	75.00	53.30	47.40	74.40	54.10	63.80
Mem0	1.1k	82.86	26.78	90.00	63.15	66.67	72.18	66.40
MemOS	1.4k	95.71	67.86	**96.67**	70.67	74.26	**77.44**	77.80
EverMemOS	2.8k	**97.14** (↑1.5%)	**85.71** (↑14.3%)	93.33 (↓3.5%)	**73.68** (↑4.3%)	**89.74** (↑20.6%)	**77.44** (↑0.0%)	**83.00** (↑6.7%)

Table 2: Main results on **LongMemEval** (accuracy, %). SS denotes single-session tasks; baselines are from the official **MemOS** results (Li et al., 2025). For EverMemOS, values in parentheses denote relative change (%) compared to the strongest baseline for that metric.

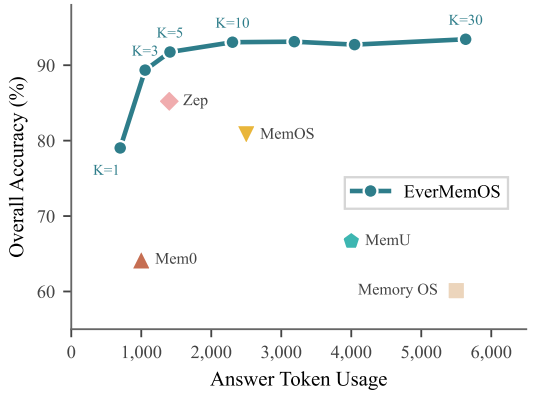

Figure 6: Performance vs. cost frontier on LoCoMo by varying the retrieved episode count (K).

(retrieval over raw dialogue). The backbone model and prompts are fixed, and only the memory representation and retrieval pipeline are varied.

As shown in Figure 4, performance degrades stepwise as structure is removed, revealing three corresponding capability losses. Removing *MemScenes* eliminates scene-level organization, weakening cross-turn aggregation over related episodes. Removing *MemCells* further drops the stable semantic units (episodes/facts), forcing retrieval to rely on raw dialogue matching. Finally, removing external memory collapses long-horizon performance, indicating that many queries cannot be handled reliably within the context window alone.

Effectiveness of Episode Segmentation. We evaluate semantic episode segmentation against fixed heuristics and ground-truth boundaries under **w/o MemScene** to isolate boundary quality. We compare three strategies: (1) **Fixed Heuristics** (fixed message count $N = 10$ or token thresholds $N = 512, 1024$); (2) **Session (Oracle)** (ground-truth session boundaries); and (3) **EverMemOS** (semantic segmentation with different backbones).

Table 3 shows that (i) semantic segmentation consistently outperforms fixed heuristics, especially coarse token chunking; (ii) it also outperforms Session (Oracle), suggesting sessions are not always optimal retrieval units; and (iii) results are robust across boundary-detection backbones (accuracy changes $\leq$0.7 points).

Summary of per-phase contributions. Taken together with the efficiency analysis in Section 4.4, the ablations above isolate the contribution of each lifecycle phase on LoCoMo. Phase I (semantic segmentation) consistently outperforms fixed heuristics and oracle session boundaries (Table 3).

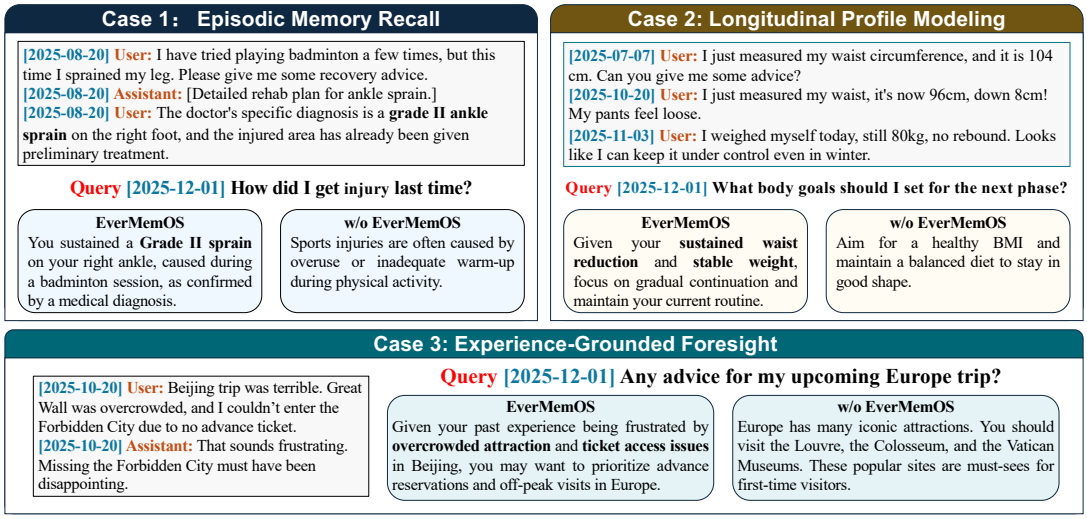

Figure 7: Case studies illustrating Profile, Foresight, and Episode capabilities in Memory-Augmented Chat.

Segmentation Method	Answer Model	
	GPT-4.1-mini	Qwen3-4B
Heuristic Baselines		
Fixed-Message-10	88.05	80.95
Fixed-Token-512	87.55	80.67
Fixed-Token-1024	84.52	75.19
Semantic Segmentation		
Session (Oracle)	87.66	80.63
Default (EverMemOS)		
w/ GPT-4.1-mini	89.16	83.07
w/ Qwen3-4B	89.78	82.73

Table 3: Comparison of boundary detection strategies. **Session (Oracle)** uses the ground-truth session partitions provided by LoCoMo.

Scenario	Ep.+Prof.	Prof.-only	Ep.-only
Consultation	**51.03**	47.33	44.44
Email (Personal)	**53.85**	46.15	46.15
Translation	**50.00**	46.15	38.08
Email (Professional)	**53.79**	41.38	45.17
Writing (Creative)	**55.10**	48.57	42.04
Writing (Professional)	**45.56**	44.79	40.15
Knowledge Query	**63.68**	62.94	54.73
Social Media	**47.90**	44.96	36.13
Chat	**52.09**	44.87	41.83
Overall	**53.25**	48.30	43.93

Table 4: Profile ablation on PersonaMem v2 (Jiang et al., 2025) (5,000 questions across 9 scenarios; accuracy, %).

Phase II (MemScene organization) yields measurable gains over flat MemCell retrieval (Figure 4). Phase III (agentic verification-and-rewriting) further improves accuracy by composing necessary and sufficient context (Section 4.4). The three phases provide complementary gains that together account for the full performance of EverMemOS. We additionally report a clustering-method ablation for Phase II in Appendix B.2.

4.4 Hyperparameter Analysis

We investigate the impact of retrieval scope via two hyperparameters: the number of retrieved *MemScenes* (N) and episodes (K). As shown in Figure 5, performance gains saturate around $N = 10$. Figure 6 further illustrates the efficiency–accuracy frontier governed by K. We therefore adopt $N = 10$ and $K = 10$ as the default configuration to balance performance with computational cost. Comprehensive sensitivity analysis is detailed in Appendix B.3.

4.5 Profile Study

We evaluate the effect of the consolidated user profile on PersonaMem-v2 (32k) (Jiang et al., 2025); results are not directly comparable across dataset versions due to differences in task setup and annotations. Table 4 shows that adding the User Profile to episodic evidence improves overall accuracy by 9.32 points over episodes-only (53.25 vs. 43.93), indicating that semantic consolidation provides complementary signal beyond episodic retrieval. We defer the full comparison against other memory systems on PersonaMem-v2 to Appendix A.5.

4.6 Case Study

Existing benchmarks primarily evaluate answer-level accuracy/recall and do not capture several capabilities required for long-term conversational agents, such as conflict detection, profile stability, and experience-grounded foresight. These capabilities are also reflected in the quantitative re-

sults. Profile stability contributes +9.32 points on PersonaMem-v2 (Table 4). Conflict detection shows a +20.6% gain on LongMemEval knowledge update (Table 2). Temporal awareness yields a +16.1% gain on LoCoMo temporal questions under GPT-4.1-mini (Table 1). To complement these numbers with richer context, Figure 7 shows three representative cases: **(Episode)** reconstructing a concrete past injury episode (a Grade-II ankle sprain during badminton) rather than producing a generic explanation; **(Profile)** maintaining longitudinal stability and using sustained improvements (waist 104→96 cm with stable weight) for trajectory-consistent goal setting; and **(Foresight)** leveraging previously observed failures (overcrowding and missing advance tickets) to make proactive recommendations for future travel. Together, these cases illustrate coherent, experience-aware behavior beyond what is measured by existing benchmarks.

5 Conclusion

In this paper, we introduced EverMemOS, a unified memory operating system for long-horizon LLM agents. By modeling an explicit memory lifecycle composed of episodic trace formation, semantic consolidation, and reconstructive recollection, EverMemOS achieves state-of-the-art performance on memory-augmented reasoning benchmarks, with particularly strong gains on multi-hop and temporal questions. We hope EverMemOS provides an extensible foundation for building more consistent and context-aware interactive agents.

Limitations

We evaluate EverMemOS on text-only conversational benchmarks. Extending the *MemCell*/*MemScene* abstraction to multimodal or embodied settings is left for future work. Current benchmarks also lack protocols for stress-testing ultra-long timelines, so our evaluation does not fully isolate performance in such regimes. This motivates specialized benchmarks for long-term memory organization and consolidation.

EverMemOS introduces LLM-mediated operations for memory construction and retrieval, which increase latency and compute relative to single-pass baselines. Caching, batching, and asynchronous execution can mitigate this, but improving end-to-end efficiency remains future work. The system also applies a uniform retrieval pipeline regardless of query complexity. Complexity-aware routing or cascade retrieval could further improve efficiency for simple queries while preserving depth for complex ones.

The multi-stage pipeline may propagate errors from early phases to downstream modules. In practice, each phase offers a degree of progressive correction: narrative synthesis in Phase I, scene-level clustering in Phase II, and the agentic sufficiency check in Phase III. Empirically, varying the Phase I boundary-detection backbone changes downstream accuracy by at most 0.7 points (Table 3), although a systematic study of cascading errors remains future work. In addition, the necessity-and-sufficiency criterion is operationalized through structured LLM prompting rather than a formal theoretical framework. Information-theoretic or decision-theoretic formulations offer a promising future direction.

Ethical Considerations

Sensitive information in user profiles. EverMemOS accumulates persistent user-specific memory, including health- and identity-related attributes that may surface in profiles. Deployments should restrict which profile fields each downstream task consumes, avoid using extracted attributes for high-stakes decisions (e.g., eligibility, pricing, access) where they may produce discriminatory outcomes, and disclose to users what categories of information the system retains.

Transparency and erasure. Unlike memory embedded in model weights, MemCells and MemScenes are stored as inspectable text and structured fields. This makes it feasible to show users what was remembered and to delete specific entries on request, which supports GDPR-style compliance. Complete erasure additionally requires refreshing derived artifacts such as scene summaries and profile fields, in order to prevent residual leakage.

Potential for misuse. Persistent memory, profiling, and foresight could also support intrusive surveillance or manipulative personalization. We release EverMemOS as research-oriented infrastructure and encourage deployers to adopt explicit consent flows, retention limits, and human oversight.

Human evaluation. Annotators for our LLM-as-judge reliability study were recruited via Prolific with informed consent and compensated at ∼$12/hour, consistent with fair-pay guidelines for academic research (Appendix A.2).

References

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Atilim Gunes Bulatov, Valentin Khrulkov, Leyla Mirvakhabova, Alexey Markov, Artem Babenko, and Ivan Oseledets. 2022. Recurrent memory transformer. In *Advances in Neural Information Processing Systems*, pages 20230–20243.

Aydar Bulatov, Yuri Kuratov, and Mikhail S. Burtsev. 2023. Scaling transformer to 1m tokens and beyond with rmt. *ArXiv*, abs/2304.11062.

Guanzheng Chen, Xin Li, Zaiqiao Meng, Shangsong Liang, and Lidong Wang. 2024. Clex: Continuous length extrapolation for large language models. In *International Conference on Learning Representations*.

Guanzheng Chen, Xin Li, Michael Qizhe Shieh, and Lidong Bing. 2025. LongPO: Long context self-evolution of large language models through short-to-long preference optimization. In *The Thirteenth International Conference on Learning Representations*.

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

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988. Association for Computational Linguistics.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Greg Slabaugh, and Tinne Tuytelaars. 2022. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385.

Mohamed Amine Ferrag, Norbert Tihanyi, and Merouane Debbah. 2025. From llm reasoning to autonomous ai agents: A comprehensive review. *arXiv preprint arXiv:2504.19678*.

Yue Gong and 1 others. 2024. M+: An efficient memory structure for large language models. *arXiv preprint arXiv:2404.09337*.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Keck, William O'Brien, Alistair Kritzman, Stanislav Illarionov, Edward Grefenstette, Tiago Wuthrich, and 1 others. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476.

Bowen Jiang, Yuan Yuan, Maohao Shen, Zhuoqun Hao, Zhangchen Xu, Zichen Chen, Ziyi Liu, Anvesh Rao Vijjini, Jiashu He, Hanchao Yu, Radha Poovendran, Gregory Wornell, Lyle Ungar, Dan Roth, Sihao Chen, and Camillo Jose Taylor. 2025. Personamem-v2: Towards personalized intelligence via learning implicit user personas and agentic memory. *arXiv preprint arXiv:2512.06688*.

Sheena A Josselyn, Stefan Köhler, and Paul W Frankland. 2015. Finding the engram. *Nature Reviews Neuroscience*, 16(9):521–534.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. 2025. Memory os of ai agent. *arXiv preprint arXiv:2506.06326*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Yuxiang Kukliansky, Wen-tau Yih Chen, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

Zhiyu Li, Shichao Song, Chenyang Xi, Hanyu Wang, Chen Tang, Simin Niu, Ding Chen, Jiawei Yang, Chunyu Li, Qingchen Yu, and 1 others. 2025. Memos: A memory os for ai system. *arXiv preprint arXiv:2507.03724*.

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

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of llm agents. *Preprint*, arXiv:2402.17753.

James L McGaugh. 2000. Memory–a century of consolidation. *Science*, 287(5451):248–251.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409. Association for Computational Linguistics.

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

Charles Packer, Vivian Woodside, Neal Dhir, and Douwe Kiela. 2024. Memgpt: Towards llms as operating systems. In *Advances in Neural Information Processing Systems*.

Ori Ram, Eyal Shnarch, Jonathan Uziel, Lisa Haklay, and Amir Globerson. 2023. In-context retrieval-augmented language models. *Transactions of the Association for Computational Linguistics*, 11:1316–1331.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. 2025. Zep: A temporal knowledge graph architecture for agent memory. *Preprint*, arXiv:2501.13956.

Daniel L Schacter. 2008. *Searching for memory: The brain, the mind, and the past*. Basic books.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36.

Haoran Sun and Shaoning Zeng. 2025. Hierarchical memory for high-efficiency long-term reasoning in llm agents. *arXiv preprint arXiv:2507.22925*.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. 2023. Longmem: Augmenting language models with long-term memory. In *Advances in Neural Information Processing Systems*, volume 36, pages 20292–20306.

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

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. 2025. Longmemeval: Benchmarking chat assistants on long-term interactive memory. *Preprint*, arXiv:2410.10813.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yi Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, and 1 others. 2023. The rise and potential of large language model based agents: A survey. *arXiv preprint arXiv:2309.07864*.

Chun Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. 2024. Agentless: Demystifying llm-based software engineering agents. *ArXiv*, abs/2407.01489.

Asaf Yehudai, Lilach Eden, Alan Li, Guy Uziel, Yilun Zhao, Roy Bar-Haim, Arman Cohan, and Michal Shmueli-Scheuer. 2025. Survey on evaluation of llm-based agents. *arXiv preprint arXiv:2503.16416*.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big Bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems*, pages 17283–17296.

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

Yuan Zheng and 1 others. 2024. Memoryllm: A framework for personalized and long-term dialogue generation. *arXiv preprint arXiv:2401.17122*.

W Zhong, L Guo, Q Gao, and 1 others. 2024. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19724–19731.

A Evaluation Details

A.1 Evaluation Settings and Fair Comparison

LoCoMo backbones. We report LoCoMo results under two backbones: GPT-4.1-mini (primary, reflecting an up-to-date backbone) and GPT-4o-mini (to facilitate comparison with prior work). Following common practice in LTMOS evaluation, we standardize the backbone used for *final answer generation* to isolate the contribution of memory management from the base model.

Baseline executability. For EverMemOS and MemoryOS, we execute the full pipeline (memory construction, retrieval, and answering) with the specified backbone. For Mem0, MemU, MemOS, and Zep, we use their official APIs for memory management/retrieval; in this setting, we keep each baseline's official memory configuration and prompting unchanged and apply the unified backbone only at the answering stage.

LongMemEval. Due to the extreme input length of LongMemEval, we cannot stably run all baseline APIs end-to-end; we therefore report baseline results from the official MemOS leaderboard[2] and evaluate EverMemOS with GPT-4.1-mini under the same protocol.

Retrieval configuration. EverMemOS uses a hybrid retriever that fuses dense retrieval (encoder: `Qwen3-Embedding-4B` (Zhang et al., 2025)) and sparse retrieval (BM25) via Reciprocal Rank Fusion (RRF), followed by episode re-ranking (`Qwen3-Reranker-4B` (Zhang et al., 2025)). Unless otherwise specified, we retrieve the top-10 *MemScenes* and select 10 Episodes for downstream inference.

MemBase statistics and construction hyperparameters. We report dataset-level MemBase statistics (Table 5) and memory-construction hyperparameters (Table 6) for LoCoMo and LongMemEval. *MemScenes* are the clustering units produced by Phase II, and each MemScene contains a small set of *MemCells*. We use the same pipeline across datasets, while adopting dataset-specific clustering hyperparameters to reflect different dialogue structures and time spans. LongMemEval contains 500 dialogue–question pairs (one per conversation). **Max time gap** is the maximum allowed temporal distance (in days): when assigning a MemCell A to a candidate MemScene, if the closest-in-time MemCell B already in that MemScene is farther than this threshold, A is not clustered into that MemScene.

[2]Leaderboard data: https://huggingface.co/datasets/MemTensor/MemOS_eval_result

Table 5: MemBase statistics on LoCoMo and LongMemEval.

Metric	LoCoMo	LongMemEval
Dataset scale		
#Conversations	10	500
#Questions	1,540	500
MemBase statistics		
#Total MemCells	702	54,755
#Total MemScenes	286	40,138
Avg MemCells/conv.	70.2	109.5
Avg MemScenes/conv.	28.6	80.3
Avg MemCells/MemScene	2.45	1.36
MemCells/conv. (range)	34–95	82–154
MemScenes/conv. (range)	13–49	60–102

Table 6: Memory-construction hyperparameters.

Hyperparameter	LoCoMo	LongMemEval
Clustering threshold τ	0.70	0.50
Max time gap (days)	7	30

Multi-round query rewriting frequency. On LoCoMo (GPT-4.1-mini), the sufficiency checker triggers a second-round query rewriting for 31.0% of questions.

Default evaluation mode. Unless otherwise specified, quantitative experiments use **Memory-Augmented Reasoning** (Episodes-only). We additionally report the effect of the consolidated **Profile** in Table 4, while **Foresight** is illustrated in the qualitative **Case Study** (**Memory-Augmented Chat**).

A.2 LLM-as-Judge Reliability

We randomly selected 25 non-overlapping Q&A pairs from LoCoMo and 25 from LongMemEval, and generated model answers for each question. We recruited annotators via Prolific. For each Q&A pair, five independent human evaluators judged whether the generated answer was correct given the question and the reference answer. All participants provided informed consent via the platform interface and were compensated at approximately $12.00/hour, consistent with fair-pay guidelines for academic research and above local minimum wage standards. Table 7 shows strong agreement between the LLM-as-judge protocol and human annotations: Cohen's κ exceeds 0.89 and accuracy remains above 98% across benchmarks. Pearson r

is 0.891 on LoCoMo and 0.979 on LongMemEval. These results suggest that GPT-4o-mini achieves human-level reliability for answer verification, enabling evaluation that is rigorous, reproducible, and cost-efficient.

Table 7: Reliability matrix for LLM-as-Judge.

Model	Cohen's κ	95% CI	Accuracy	Pearson r
LoCoMo	0.891	[0.742, 1.000]	0.984	0.891
LongMemEval	0.978	[0.936, 1.000]	0.992	0.979

A.3 Token Cost Breakdown

To improve cost transparency, we log *all* LLM API calls during LoCoMo evaluation (1,540 questions) under two backbones (GPT-4.1-mini and GPT-4o-mini) and attribute token usage to stages in our pipeline. Since LoCoMo evaluation uses **Memory-Augmented Reasoning** (Episodes-only), we do *not* invoke the **Profile** module; therefore, profile-related tokens are excluded from Table 8. Table 8 maps stages to EverMemOS phases. Phase I corresponds to memory construction (`add`). In this Episodes-only setting, Phase II uses non-LLM computation (clustering/embedding updates) and thus incurs no additional LLM tokens. Phase III consists of retrieval (`search`) and answer generation (`answer`). The `evaluate` stage reflects LLM-as-judge scoring (three judges per question) and is reported separately. Phase III consumes 10.27M tokens (∼6.7k/question) with GPT-4.1-mini and 9.31M tokens (∼6.0k/question) with GPT-4o-mini; Phase I consumes 9.42M and 9.34M tokens, respectively, amortized over memory building.

Table 8: Token-level cost breakdown on LoCoMo (1,540 questions) under two backbones. Tokens are reported in millions (M); Total includes both prompt and completion.

Stage	#Calls	Prompt (M)	Total (M)
GPT-4.1-mini			
add	7056	8.66	9.42
search	2017	4.12	4.45
answer	1540	4.63	5.82
search+answer	**3557**	**8.75**	**10.27**
evaluate	4620	2.35	2.38
GPT-4o-mini			
add	7250	8.60	9.34
search	2219	4.37	4.62
answer	1540	3.84	4.69
search+answer	**3759**	**8.21**	**9.31**
evaluate	4620	2.14	2.17

A.4 Runtime and Storage Cost

Beyond token usage, we also report system-level overhead to assess practical feasibility. Table 9 summarizes per-query LLM calls, end-to-end latency, and per-MemCell storage in our default deployment. Each query triggers two LLM calls in the default path: one sufficiency check and one answer generation. When the verifier judges the retrieved context insufficient, the agentic rewriting loop adds one or two additional calls. End-to-end inference latency is 3–5 seconds per query. Storage averages ∼20 KB per MemCell, covering raw text, atomic facts, foresight signals, and metadata. Since clustering and retrieval are stateless passes over an append-only store (Section 3.4), the system supports horizontal scaling without reprocessing existing memory.

Table 9: Runtime and storage cost per query / MemCell.

Dimension	Value
LLM calls (default, one-shot)	2
LLM calls (with agentic rewriting)	3–4
End-to-end inference latency	3–5 s
Storage per MemCell	∼20 KB

A.5 PersonaMem v2: Full Comparison Results

Table 10 reports the full comparison on PersonaMem v2 (32k) (Jiang et al., 2025) (2,447 questions across 9 scenarios). The *Profile* row indicates whether a memory system provides a profile-like component (not necessarily named "Profile") that summarizes stable user information (e.g., MemOS maintains explicit vs. implicit preferences). For methods with such a component (✓), we generate answers using the retrieved memories *plus* the system's profile-like component; for methods without it (✗), we generate answers using the retrieved memories only. EverMemOS achieves the best overall accuracy (53.25%), outperforming the strongest baseline (MemOS, 50.72%) by 2.53 points.

B Additional Analyses

B.1 MemScene Definition and Online Construction

We provide the complete definition and online construction procedure for MemScenes introduced in Section 3.4. A **MemScene** is a triple $S = (\mathcal{C}, \mathbf{e}_{\text{centroid}}, t_{\text{last}})$:

Scenario	Zep	Mem0	MemU	MemoryOS	MemOS	EverMemOS
Consultation	39.51	43.21	37.86	35.80	48.15	**51.03**
Email (Personal)	42.51	41.30	33.20	36.84	49.80	**53.85**
Translation	36.92	43.08	38.46	40.00	**51.92**	50.00
Email (Professional)	37.59	42.41	32.76	35.86	50.00	**53.79**
Creative Writing	41.22	42.86	35.51	35.51	48.16	**55.10**
Writing (Professional)	40.54	34.75	35.14	35.91	**48.26**	45.56
Knowledge Query	63.43	59.20	56.97	57.96	61.94	**63.68**
Social Media	32.35	38.66	34.03	35.29	46.64	**47.90**
Chat	44.87	40.30	34.22	36.88	44.87	**52.09**
Profile	✗	✗	✓	✓	✓	✓
Overall	43.40	43.85	38.70	40.05	50.72	**53.25**

Table 10: Full comparison on PersonaMem v2 (32k) (Jiang et al., 2025) (5,000 questions across 9 scenarios; accuracy, %).

- $\mathcal{C} = \{c_1, \ldots, c_k\}$: the set of member MemCells. Each MemCell belongs to exactly one MemScene.
- $\mathbf{e}_{\text{centroid}}$: the running-mean embedding, incrementally updated as $\mathbf{e}_{\text{centroid}} \leftarrow \frac{(|\mathcal{C}|-1)\,\mathbf{e}_{\text{centroid}}+\mathbf{e}(c)}{|\mathcal{C}|}$ upon assimilating a new MemCell c.
- t_{last}: the timestamp of the most recently added MemCell, used for temporal gating.

Online construction. When a new MemCell c arrives at time t_c:

1. Compute its embedding $\mathbf{e}(c)$.
2. Retrieve all candidate MemScenes whose temporal gap satisfies $t_c - t_{\text{last}}(S) \leq \Delta_{\max}$.
3. Among the candidates, find the MemScene S^* with the highest cosine similarity $\text{sim}(\mathbf{e}(c), \mathbf{e}_{\text{centroid}}(S))$.
4. If sim $> \tau$ and no profile-level conflict is detected, assimilate c into S^* and update $\mathbf{e}_{\text{centroid}}$ and t_{last}; otherwise, instantiate a new MemScene $S_{\text{new}} = (\{c\}, \mathbf{e}(c), t_c)$.

This greedy online procedure processes each MemCell exactly once with amortized cost $O(|\mathcal{S}|)$ per cell, where $|\mathcal{S}|$ is the current number of MemScenes. No batch re-clustering is required. In our experiments, we set τ=0.70 and $\Delta_{\max}$=7 days for LoCoMo, and τ=0.50 and $\Delta_{\max}$=30 days for LongMemEval (see Table 6).

B.2 Clustering Method Ablation

To evaluate the quality of MemScene construction, we compare four clustering approaches on all 10 LoCoMo conversations (∼71 MemCells each), measured against ground-truth session boundaries. We consider: (1) **Emb. Cosine** (our method), (2) **Time-only** (merge by temporal adjacency), (3) **BM25** (normalized BM25 similarity), and (4) **Jaccard** (token-level set overlap).

Table 11: Clustering quality against session boundaries on LoCoMo.

Method	NMI	ARI	Homo.	Comp.	V-meas.
Emb. Cosine (ours)	0.829	0.288	**0.847**	0.819	0.829
Time-only	0.831	0.386	0.714	**1.000**	0.831
BM25	0.558	0.043	0.482	0.686	0.558
Jaccard	0.440	0.023	0.331	0.679	0.440

As shown in Table 11, our embedding-cosine method achieves the highest Homogeneity (0.847), indicating that individual MemScenes are thematically pure. Time-only clustering achieves perfect Completeness (1.000) by grouping all temporally adjacent cells, but this comes at the cost of mixing distinct topics within the same cluster.

Table 12: Scene purity analysis on LoCoMo.

Method	Intra	Inter	Sep.	#Scenes	Size
Emb. Cosine (ours)	0.740	0.733	**+0.007**	41.1	1.84
Time-only	0.716	0.864	−0.147	13.3	5.53
Jaccard	0.720	0.781	−0.061	12.7	5.96
BM25	0.713	0.764	−0.052	21.3	3.55

Table 12 further reports scene purity via the Separation metric (intra-scene similarity minus inter-scene similarity). Only our method achieves positive Separation (+0.007), confirming that MemScenes are internally more coherent than they are similar to other scenes. All other methods yield negative Separation, indicating frequent topic leakage across scenes.

B.3 Hyperparameter Sensitivity and Efficiency Trade-off

To better understand retrieval budgets, we analyze the MemScene budget N and episode budget K under a simplified setting that disables the agentic verification-and-rewriting loop in Phase III, isolating one-shot retrieval. Figure 5 shows that increasing N improves evidence-session recall and answer accuracy initially but quickly saturates; $N{=}10$ already yields strong recall. We therefore avoid brute-force expansion of the retrieved scene set for efficiency. We also set $N{=}10$ to ensure the candidate pool contains at least $K{=}10$ MemCells even in extreme cases where each retrieved MemScene contains only a single MemCell. We choose $K{=}10$ episodes because most memory questions can be answered with a compact set of episodes while still covering difficult instances whose annotated evidence spans up to 7–8 recalled episodes. Finally, Figure 6 shows a favorable cost–accuracy frontier: decreasing K substantially reduces tokens used for downstream reasoning, and at moderate K values EverMemOS can achieve both lower token usage and higher accuracy than strong baselines.

B.4 Accuracy Exceeding Recall on LoCoMo

In Figure 5, accuracy can exceed recall at small K on LoCoMo. Table 13 quantifies this effect: even when none of the *annotated* evidence sessions are retrieved ("zero recall"), 12–20% of questions are still answered correctly.

Table 13: Accuracy vs. recall statistics on LoCoMo.

Metric	$K{=}1$	$K{=}3$
Recall	65.06%	86.32%
Accuracy	71.80%	87.81%
Zero-recall questions	429	125
Answered correctly	52 (12.1%)	25 (20.0%)

This primarily reflects **information redundancy** and **non-unique evidence annotations**: salient facts (identity, preferences, goals) recur across sessions, so the annotated evidence is not always the only session that supports the answer. For example, a question about "Caroline's identity" is annotated with session [1], yet sessions [11–15] also state she is a transgender woman, enabling a correct answer from alternative sessions. In addition, LLMs can sometimes infer the correct response from semantically related retrieved content even when the exact annotated session is missing.

Overall, recall computed against annotated evidence can underestimate retrieval usefulness when evidence is distributed. Increasing K from 1 to 3 reduces zero-recall cases by 71% (429→125), narrowing the accuracy–recall gap.

Illustrative Cases. We provide three representative examples where answers remain correct despite missing the annotated evidence sessions:

- **Redundant identity facts.** *Q: "What is Caroline's identity?"* The gold answer is *transgender woman*. Although the evidence is annotated in session [1], later sessions also explicitly mention this identity; the retriever surfaces those alternatives at small K, and the model answers correctly.
- **Distributed activity mentions.** *Q: "What activities does Melanie partake in?"* The gold answer spans multiple hobbies (e.g., pottery, camping, painting, swimming) with evidence annotated across multiple sessions. Retrieved sessions may miss the annotated ones but still contain sufficient mentions (e.g., pottery/painting) to support a correct response.
- **Inference from related signals.** *Q: "Would Caroline pursue writing as a career option?"* While the evidence is annotated in session [7], retrieved content from other sessions describes her career goal (e.g., becoming a counselor), enabling the LLM to infer that writing is unlikely.

B.5 Profile Extraction Example

EverMemOS maintains a compact **User Profile** with two fields: *explicit facts* (verifiable attributes and time-varying measurements) and *implicit traits* (preferences and habits). The profile is updated online from Phase II scene summaries with recency-aware updates for time-varying fields and conflict tracking when evidence is inconsistent. Table 14 provides an abridged example.

C Reproducibility Artifacts

C.1 Prompts for Agentic Retrieval

To make our system behavior transparent and reproducible, we include the core prompt templates used by our agentic retrieval controller.[3]

[3]Our implementation is open-sourced; we still include prompts here to keep the paper self-contained for review.

Table 14: Profile extraction example (de-identified): abridged evidence snippets and the resulting user profile.

Evidence snippets (excerpt)	Retrieved user profile (excerpt)
2025-07-07: "I just measured my waist circumference, and it is 104 cm. Can you give me some advice?" **2025-10-20**: "My waist is now 96 cm, down 8 cm! My pants feel loose." **2025-11-03**: "The doctor said my fatty liver has improved (moderate → mild). Waist is now 95 cm." **2025-11-03**: "My weight is still 80 kg, no rebound. I can keep it under control even in winter."	**Explicit facts.** Waist circumference: baseline 104 cm; latest 95 cm ($\Delta = -9\,\mathrm{cm}$). Weight: stable at 80 kg (no rebound). Fatty liver grade: moderate → mild (improved). **Implicit traits.** Self-management: goal-oriented; consistently tracks health metrics and responds well to feedback. Preference: requests immediately actionable adjustments.

Sufficiency check. We use an LLM-based sufficiency check to decide whether the currently retrieved documents contain enough evidence to answer the user query. The prompt template (with placeholders) is shown below.

```
You are an expert in information retrieval
↪  evaluation. Assess whether the retrieved
↪  documents provide sufficient information to
↪  answer the user's query.

User Query:
{query}

Retrieved Documents:
{retrieved_docs}

### Instructions:

1. **Analyze the Query's Needs**
   - **Entities**: Who/What is being asked about?
   - **Attributes**: What specific details
   ↪  (color, time, location, quantity)?
   - **Time**: Does it ask for a specific time
   ↪  (absolute or relative like "last week")?

2. **Evaluate Document Evidence**
   - Check **Content**: Do the documents mention
   ↪  the entities and attributes?
   - Check **Dates**:
     - Use the `Date` field of each document.
     - For relative time queries (e.g., "last
     ↪  week", "yesterday"), verify if document
     ↪  dates fall within that timeframe.
     - If the query asks "When did X happen?", do
     ↪  you have the specific date or just a
     ↪  vague mention?

3. **Judgment Logic**
   - **Sufficient**: You can answer the query
   ↪  *completely* and *precisely* using ONLY
   ↪  the provided documents.
   - **Insufficient**:
     - The specific entity is not found.
     - The entity is found, but the specific
     ↪  attribute (e.g., "price") is missing.
     - The time reference cannot be resolved
     ↪  (e.g., doc says "yesterday" but has no
     ↪  date, or doc date doesn't match query
     ↪  timeframe).
     - Conflicting information without
     ↪  resolution.

### Output Format (strict JSON):

{{
  "is_sufficient": true or false,
  "reasoning": "Brief explanation. If
  ↪  insufficient, state WHY (e.g., 'Found X but
  ↪  missing date', 'No mention of Y').",
  "key_information_found": ["Fact 1 (Source: Doc
  ↪  1)", "Fact 2 (Source: Doc 2)"],
  "missing_information": ["Specific gap 1",
  ↪  "Specific gap 2"]
}}

Now evaluate:
```

Multi-query generation (condensed). When the current retrieval is deemed insufficient, we generate 2–3 complementary follow-up queries targeted at the missing information. We omit examples and keep only the constraints that affect behavior (inputs, strategy choices, and the strict JSON output schema).

```
You are an expert at query reformulation for
↪  conversational memory retrieval.
Your goal is to generate 2-3 complementary
↪  queries to find the MISSING information.

--------------------------
Original Query:
{original_query}

Key Information Found:
{key_info}

Missing Information:
{missing_info}

Retrieved Documents (Context):
{retrieved_docs}
--------------------------

### Strategy Selection (choose based on why info
↪  is missing)
- Pivot / Entity Association: search related
↪  entities/categories
- Temporal Calculation: anchor relative times
↪  using document dates
- Concept Expansion: synonyms / general-specific
↪  variants
- Constraint Relaxation: remove one constraint at
↪  a time

### Query Style Requirements (use DIFFERENT
↪  styles)
```

```
1) Keyword Combo (2-5 words)
2) Natural Question (5-10 words)
3) Hypothetical Statement (HyDE, 5-10 words)

### Output Format (STRICT JSON)
{
  "queries": ["Query 1", "Query 2", "Query 3"],
  "reasoning": "Strategy used for each query
  ↪  (e.g., Q1: Pivot, Q2: Temporal)"
}
```

C.2 End-to-End Inference Trace (LoCoMo Multi-Hop Example)

To improve transparency, we provide an end-to-end inference trace for a representative LoCoMo multi-hop question (conversation `locomo_6`), including the MemBase hierarchy (*MemScenes* and *MemCells*) and the two-round retrieval process (sufficiency check and query rewriting) that leads to a correct final answer. We denote the retrieved MemScene count as N and the retrieved MemCell (episode) count as K (corresponding to `scene_top_k` and `response_top_k` in our implementation).

Trace at a glance.

- **Question (multi-hop).** *"Does James live in Connecticut?"* The dialogue never directly states James's residence; the system must infer the answer from related evidence.
- **MemBase hierarchy.** 49 MemScenes / 91 MemCells; retrieval selects top N=10 MemScenes (20%), then reranks/selects K=10 MemCells for answering.
- **Round 1 retrieval + sufficiency.** Top N=10 MemScenes (31 MemCells) $\rightarrow$ insufficient (`is_sufficient=false`); missing an explicit residence mention / confirmation of living in Connecticut.
- **Query rewriting.** The controller generates refined queries targeting residence/location information.
- **Round 2 retrieval.** With 40 additional candidates, the top-ranked MemCell contains the key evidence that James adopted a dog from a shelter in Stamford, enabling an evidence-grounded inference.
- **Inference + evaluation.** Final answer: *Likely yes*; judged correct by 3/3 LLM judges.

Worked example (formatted). For readability, we summarize the trace in Table 15 (instead of printing raw JSON).

Table 15: End-to-end inference trace (LoCoMo multi-hop example), summarized.

Stage	Key outputs
Input	Query: *Does James live in Connecticut?* (Category: multi-hop; Gold: *Likely yes*).
MemBase	49 MemScenes / 91 MemCells (conversation `locomo_6`).
Round 1	Top N=10 MemScenes (31 MemCells) $\rightarrow$ insufficient (`is_sufficient=false`); missing an explicit residence mention / confirmation of Connecticut.
Rewrite	Refined queries: (i) `James residence Connecticut`; (ii) `Where does James currently live`; (iii) `James lives near McGee's bar in Connecticut`.
Round 2	+40 candidates; top result is *James Adopts Shelter Dog Ned... (Apr 12, 2022)* from `cluster_004`, mentioning "Stamford".
Answer	Output: *Likely yes*; judged correct by 3/3 LLM judges.

Detailed trace. Round 1: initial retrieval and sufficiency check.

- **Retrieval mode.** Agentic MemScene-guided reranking (`agentic_scene_rerank`) with N=10 and K=10.
- **Retrieved candidates.** N=10 MemScenes (31 MemCells).
- **Sufficiency verdict.** `is_sufficient=false`.
- **Key information found.** "James and Samantha moved in together near McGee's Bar"; "James traveled to Nuuk recently".
- **Missing information.** (i) explicit mention of James's residence location; (ii) confirmation whether James lives in Connecticut.

Verifier rationale (excerpt).

```
None of the documents explicitly mention where
↪  James currently lives or whether he lives in
↪  Connecticut.
```

Query rewriting (Round 2 queries).

```
1) James residence Connecticut
2) Where does James currently live
3) James lives near McGee's bar in Connecticut
```

Round 2: top retrieved MemCells. The refined queries retrieve 40 additional candidates; top results are:

Rank	Scene	Subject (summary)
1	4	**James Adopts Shelter Dog Ned and Discusses Gaming Progress with John (Apr 12, 2022)**
2	43	James Announces Move-In with Samantha Near McGee's Bar (Oct 31, 2022)
3	15	James Shares Recent Life Challenges and Support System (Jun 16, 2022)
4	44	James's Family Road Trip Visiting Friends Josh and Mark (Nov 7, 2022)
5	22	James's Upcoming Summer Trip to Toronto and Vancouver (Jul 9, 2022)

Key evidence (dialogue excerpt).

```
I adopted a pup from a shelter in Stamford last
↪  week and my days have been so much happier
↪  with him in the fam.
```

Key evidence (atomic fact).

- James adopted a pup named Ned from a shelter in Stamford last week (March 28–April 3, 2022).

Answer + evaluation.

```
James adopted his dog Ned from a shelter located
↪  in Stamford, which is a city in Connecticut,
↪  strongly implying he lives near Stamford,
↪  Connecticut. Furthermore, when James and
↪  Samantha moved in together on October 31,
↪  2022, they rented an apartment near McGee's
↪  bar, a location they both favored. Although
↪  the exact city of this apartment is not
↪  explicitly stated, given no mention of
↪  relocation to another state and the previous
↪  connection to Stamford, it is highly likely
↪  that their residence is still in Connecticut,
↪  near Stamford. Therefore, based on these
↪  details, James does live in Connecticut with
↪  high confidence.
```

Result: `is_correct=true` (3/3 judges).