Chain-of-Memory: Lightweight Memory Construction with Dynamic Evolution for LLM Agents

Xiucheng Xu[1,2,3], **Bingbing Xu**[1,2*], **Xueyun Tian**[1,2,3], **Zihe Huang**[1,2,3], **Rongxin Chen**[1,2,3], **Yunfan Li**[1,2,3], **Huawei Shen**[1,2,3]
[1]State Key Laboratory of AI Safety, Beijing, 100086
[2]Institute of Computing Technology, Chinese Academy of Sciences
[3]University of Chinese Academy of Sciences
{xuxiucheng24s,xubingbing}@ict.ac.cn

Abstract

External memory systems are pivotal for enabling Large Language Model (LLM) agents to maintain persistent knowledge and perform long-horizon decision-making. Existing paradigms typically follow a two-stage process: computationally expensive memory construction (e.g., structuring data into graphs) followed by naive retrieval-augmented generation. However, our empirical analysis reveals two fundamental limitations: complex construction incurs high costs with marginal performance gains, and simple context concatenation fails to bridge the gap between retrieval recall and reasoning accuracy. To address above challenges, we propose **CoM (Chain-of-Memory)**, a novel framework that advocates for a paradigm shift toward lightweight construction paired with sophisticated utilization. CoM introduces a *Chain-of-Memory* mechanism that organizes retrieved fragments into coherent inference paths through dynamic evolution, utilizing adaptive truncation to prune irrelevant noise. Extensive experiments on the LongMemEval and LoCoMo benchmarks demonstrate that CoM outperforms strong baselines with accuracy gains of 7.5%–10.4%, while drastically reducing computational overhead to approximately 2.7% of token consumption and 6.0% of latency compared to complex memory architectures. Code is available at https://github.com/Xiucheng-Xu/CoM.

1 Introduction

Large Language Model (LLM)-driven agents are evolving from simple conversational interfaces into autonomous entities capable of handling complex, long-horizon tasks (Yang et al., 2024; Singh et al., 2025; Zhang et al., 2025a). Effective decision-making in such scenarios necessitates the continuous integration of extensive interaction histories. However, the inherent constraints of finite context windows (Fei et al., 2024; Zhang et al., 2025c) in LLMs reduce their ability to retain vast amounts of information, which limits their potential for long-term knowledge accumulation and adaptability (Wang et al., 2024; Liu et al., 2024). As a result, equipping agents with explicit external memory systems (Zhong et al., 2024; Li et al., 2025) has emerged as a critical component for enabling persistent knowledge accumulation and long-horizon decision-making in LLM-based agents.

Existing works on agent memory generally adopt a two-stage paradigm: memory construction followed by memory utilization. During construction, raw interaction traces are often transformed into structured formats, such as trees or graphs, to capture semantic connections (Chhikara et al., 2025; Xu et al., 2025), resulting in high computational costs. Conversely, memory utilization typically adopts naive Retrieval-Augmented Generation (RAG) (Gao et al., 2023), a conventional retrieve-and-concatenate paradigm where retrieved fragments are directly incorporated into prompts.

However, as Fig. 1 illustrated, we find that the above methods suffer from two fundamental limitations: First, our empirical observations reveal that the heavy cost incurred by elaborate memory construction is not matched by commensurate performance gains. Specifically, on long-term memory QA benchmarks, such structured memories lead to significantly higher token usage and latency, while yielding negligible accuracy improvements over a naive RAG baseline. Second, there exists a clear gap between the retrieval effectiveness and final response accuracy: even when ground-truth evidence is successfully retrieved, directly injecting retrieved fragments into the prompt often fails to translate recall into accurate answers, suggesting that the prevailing practice of merely inserting fragments into prompts is insufficient for reasoning.

Motivated by these observations, we argue for a paradigm shift in memory design: rather

* Corresponding author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 11618–11631

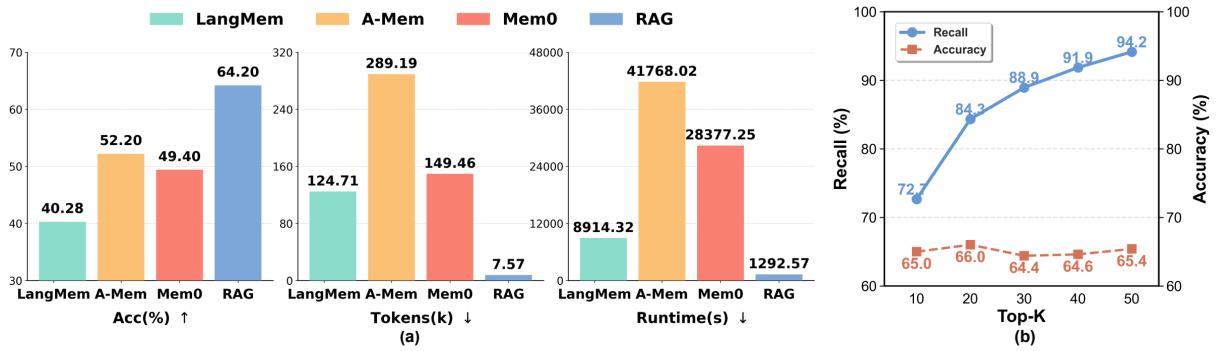

Figure 1: **Empirical limitations of existing paradigms.** (a) Heavy-weight memory construction strategies fail to demonstrate cost-effectiveness. (b) Naive retrieval strategies exhibit a reasoning bottleneck, where retrieved evidence is not effectively utilized for answer generation.

than investing in costly and elaborate memory construction coupled with naive retrieve-and-concatenate utilization, memory systems should adopt lightweight construction with more principled and effective utilization. This shift requires rethinking how retrieved memories are organized and consumed during generation. In particular, we identify two key limitations in existing memory utilization strategies. First, retrieved memories are processed largely in isolation, without explicit relational structures to support compositional or multi-hop reasoning. Second, relevant evidence is often embedded within long and noisy contexts, where excessive and irrelevant information obscures critical signals, making it difficult for the model to effectively leverage retrieved memories.

Inspired by these, we propose a Chain-of-Memory framework, named CoM, featuring lightweight memory construction for retrieval and a dynamic memory chain evolution mechanism for organizing isolated fragments into coherent inference paths. Specifically, we construct these memory chains by jointly evaluating query relevance and contextual consistency. Furthermore, to mitigate information redundancy, we adopt an adaptive truncation mechanism that prunes non-essential contexts, preventing critical information from being diluted. Extensive experiments on the LongMemEval (Wu et al., 2024) and LoCoMo (Maharana et al., 2024) benchmarks demonstrate that CoM consistently outperforms the strongest baselines, yielding absolute accuracy gains of **7.5%–10.0%** with GPT-4o-mini and **9.1%–10.4%** with Qwen3-32B. Meanwhile, CoM reduces the computational footprint drastically, requiring approximately **2.7%** of the token consumption and merely **6.0%** of the time compared to prevailing complex memory architectures.

Our contributions can be summarized as follows:

- **Paradigm-Shifting:** We advocate for a fundamental shift away from costly, structured memory construction coupled with naive retrieval, towards a principle of lightweight construction paired with systematic and dynamic memory utilization.
- **Methodologically Novel:** We propose the Chain-of-Memory framework, which organizes retrieved fragments into coherent inference paths through dynamic memory chain evolution and prunes non-essential contexts via adaptive truncation.
- **Empirically Effective:** The framework delivers substantial accuracy gains (7.5%–10.4%) while drastically reducing computational costs (to 2.7% token usage and 6.0% runtime) compared to existing complex memory structures.

2 Related Work

2.1 Retrieval-Augmented Generation

Standard Retrieval-Augmented Generation (RAG) systems (Gao et al., 2023; Yu et al., 2024) partition external corpora into discrete segments, retrieving relevant chunks based on semantic similarity to augment LLM prompts. Prevailing chunking strategies include rule-based methods that generate fixed-size segments (Sarthi et al., 2024; Liu et al., 2025b), semantic-based approaches that cluster content by topic (Qu et al., 2025), and LLM-driven techniques that leverage parametric knowledge for segmentation (Pan et al., 2025; Duarte et al., 2024; Zhao et al., 2024b). Some approaches adopt GraphRAG systems (Guo et al., 2024; Dong et al., 2025) to

enhance recall through extensive relationship precomputation. However, these frameworks operate on static data and lack mechanisms to maintain an evolving memory of historical interactions. In contrast, agent memory systems are grounded in environmental interaction, dynamically assimilating information from agent actions and environmental feedback into a persistent memory store (Wang et al., 2025; Zhao et al., 2024a; Sun et al., 2025).

2.2 Memory for LLM Agents

Memory systems enable LLM-driven agents to maintain persistent context and perform consistent decision-making in complex environments (Liu et al., 2025a; Mei et al., 2025). Early approaches typically organize historical experiences as linear sequences, occasionally augmented with hierarchical structures. For instance, MemGPT (Packer et al., 2023) draws inspiration from operating system virtual memory to manage context paging, while frameworks like SCM (Wang et al., 2023) employ controller-based mechanisms to optimize information retention. While implementationally efficient, these linear methods often struggle to capture explicit semantic dependencies between temporally distant fragments. To address this, some approaches organize memory into complex structures, such as trees or graphs. Systems such as MemTree (Rezazadeh et al., 2024), Zep (Rasmussen et al., 2025), A-Mem (Xu et al., 2025), Mem0 (Chhikara et al., 2025) construct temporal knowledge trees, graphs or networks to permit dynamic updates and relational retrieval.

Although these structured approaches significantly improve reasoning consistency, they typically incur high computational costs during the memory construction phase, making them less practical for real-time applications. Furthermore, despite the sophisticated organization, the utilization stage frequently relies on naive retrieval paradigms that fail to support compositional reasoning. Diverging from these heavy-construction frameworks, our CoM proposes a paradigm shift: minimizing construction overhead while enhancing utilization through a dynamic chain of memory evolution chain-of-evidence mechanism.

3 Methodology

We propose CoM, a framework that integrates a lightweight flat index with a dynamic post-retrieval chaining mechanism. Instead of relying on complex pre-built structures, CoM constructs query-oriented dependencies among memory nodes to synthesize coherent reasoning paths. As illustrated in Figure 2, the framework comprises two core stages: (1) Memory Construction and Retrieval, and (2) Dynamic Memory Chain Evolution.

3.1 Memory Construction and Retrieval

To minimize computational overhead and temporal latency, we adopt a lightweight, flat memory architecture that stores raw conversation data without complex pre-structuring. By leveraging direct semantic similarity for retrieval, this streamlined approach ensures rapid access to relevant context, avoiding the heavy computational burden associated with hierarchical or graph-based modeling.

Construction. We formalize the raw conversation history as $\mathcal{H} = \{S_1, \ldots, S_M\}$. Each session $S_i = (t_{i,1}, \ldots, t_{i,N_i})$ consists of a chronological sequence of N_i turns, where the turn $t_{i,j}$ is a single utterance generated by either the user or the assistant. To facilitate efficient indexing, we treat a single turn as the atomic granularity for memory construction . Specifically, each turn $t_{i,j}$ is transformed into a Memory Node:

$$m_{i,j} = (x, \tau, \rho, \mathbf{e}), \tag{1}$$

where x denotes the raw textual content, τ is the timestamp, $\rho \in \{\text{User}, \text{Assistant}\}$ indicates the speaker role, and $\mathbf{e} \in \mathbb{R}^d$ is the embedding of x. We define the resulting memory database as $\mathcal{M} = \{(m_{i,1}, \ldots, m_{i,N_i})\}_{i=1}^{M}$ and simplify this notation to $\mathcal{M} = \{m_1, \ldots, m_n\}$. This organization preserves the raw structure of the original conversation, supporting both context-aware filtering and efficient retrieval of atomic memory nodes.

Retrieval. For a given query q, we employ an embedding model to obtain its vector representation $\mathbf{q} = E(q)$, and compute the relevance score for each memory node m_i via cosine similarity:

$$s_i = \cos(\mathbf{q}, \mathbf{e}_i). \tag{2}$$

The Top-K memory nodes with the highest scores are selected to form a candidate pool $\mathcal{P} = \{m_1, m_2, \ldots, m_K\} \subset \mathcal{M}$. Although $\mathcal{P}$ provides the necessary context, memory nodes within this pool remain isolated, lacking explicit logical connections to guide the reasoning process.

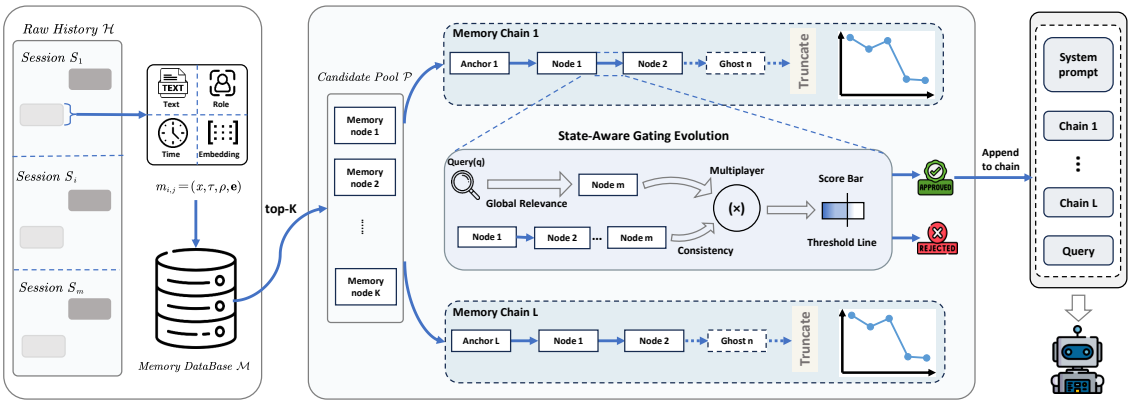

Figure 2: **The overview architecture of CoM.** The workflow consists of two stages: (1) Memory Construction and Retrieval, and (2) Dynamic Memory Chain Evolution. The dashed Ghost n denotes a candidate node whose gating score sharply drops below the adaptive threshold, thereby triggering path truncation.

3.2 Dynamic Memory Chain Evolution

To construct coherent and contextually relevant reasoning paths from the retrieved memories, we propose a Dynamic Memory Chain Evolution mechanism. This stage iteratively expands memory chains by selecting subsequent nodes that adhere to the user's original intent, while maintaining logically contextual coherence with the ongoing reasoning path. By dynamically evolving these chains, we model long-range dependencies while pruning irrelevant information which ensures high quality context for reasoning.

Initialization. The process commences by selecting the Top-L memory nodes from the candidate pool $\mathcal{P}$ according to their retrieval scores to serve as starting anchors. Consequently, L distinct memory chains are constructed, each initialized as $\mathcal{C}_z^{(0)} = \{m_z\}$. These initial anchors serve as the foundation for the subsequent iterative expansion, where each chain evolves step-by-step to incorporate complementary information.

State-Aware Gating Evolution. At expansion step t for a specific chain $\mathcal{C}_z^{(t)}$, we aim to identify the optimal successor from the candidates. To strictly control chain growth quality, we compute a gating score S_{gate} for each candidate m:

$$S_{\text{gate}}(m) = \underbrace{\cos(\mathbf{m}, \mathbf{q})}_{\text{Global Relevance}} \times \underbrace{\cos(\mathbf{m}, \mathbf{C}_z^{(t)})}_{\text{Contextual Consistency}}, \quad (3)$$

where $\mathbf{C}_z^{(t)}$ denotes the embedding of chain $\mathcal{C}_z^{(t)}$ at step t. This multiplicative formulation acts as a soft logical gate, ensuring that a valid successor satisfies two essential criteria: alignment with the user's query (Global Relevance) and coherence with the current reasoning context (Contextual Consistency). Consequently, the candidate m^* maximizing S_{gate} is selected as the next node.

Adaptive Path Truncation. To mitigate the risk of semantic drift—where the chain diverges into irrelevant topics, we implement a relative threshold termination strategy. Let s_{t-1} denote the gating score of the node appended in the previous step, and s_t^* be the optimal score among candidates in the current step. The expansion terminates if the relevance exhibits a sharp attenuation:

$$s_t^* < \beta \cdot s_{t-1}, \quad (4)$$

where $\beta \in [0, 1]$ is a hyperparameter controlling the sensitivity of the truncation. This condition identifies "cliff-like" drops in semantic relevance, indicating that further expansion would yield diminishing returns. Thus, we finalize the chain at this point to preserve the integrity and quality of the reasoning path.

3.3 Complexity Analysis

We analyze the complexity of the dynamic memory chain evolution process. Let K be the size of the retrieved candidate pool, L the number of initialized memory chains, and d the embedding dimension. Within each chain, selected candidates are removed from that chain's candidate set, while different chains may still reuse the same candidates. If adaptive truncation is never triggered, a single chain evaluates at most $\sum_{t=1}^{K-1}(K-t)$ can-

Dataset	Category	# Questions	
		Count	Ratio
LongMemEval	Single-hop (S-hop)	156	31.2%
	Multi-hop (M-hop)	133	26.6%
	Temporal (Temp)	133	26.6%
	Knowledge (Kno)	78	15.6%
LoCoMo	Single-hop (S-hop)	841	54.6%
	Multi-hop (M-hop)	282	18.3%
	Temporal (Temp)	321	20.8%
	Knowledge (Kno)	96	6.2%

Note: LongMemEval 'Single-hop' aggregates all single-session subtypes. LoCoMo excludes 446 'Adversarial' samples.

Table 1: Distribution of unified question categories for LongMemEval and LoCoMo.

didates, resulting in $O(K^2d)$ vector similarity operations. Across L independently evolved chains, the worst-case complexity is therefore $O(LK^2d)$. If a maximum chain length T is imposed, the bound becomes $O(LTKd)$, where $T \leq K$.

In practice, both L and K are deliberately kept small, and adaptive truncation usually stops chain expansion before the worst case is reached. Moreover, query-candidate relevance scores can be precomputed once for the retrieved pool, so the main repeated computation comes from contextual consistency scoring during chain evolution. As shown by the end-to-end runtime results, this overhead remains close to that of naive RAG and is substantially lower than methods that require expensive graph or tree memory construction.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate our method on two long-term memory benchmarks: LongMemEval (specifically the LongMemEval-S split) (Wu et al., 2024) and LoCoMo (Maharana et al., 2024). LongMemEval consists of 500 QA pairs derived from 500 distinct conversation histories, with an average context length of approximately 115k tokens. LoCoMo contains 1,986 QA instances from 10 conversation sets, averaging approximately 26k tokens per context. To maintain consistency across benchmarks, we map the diverse original question types into four unified categories: *single-hop*, *multi-hop*, *temporal reasoning*, and *knowledge*. Following Mem0 (Chhikara et al., 2025), we exclude the *adversarial* category from LoCoMo to ensure fair comparison with baselines. Table 1 presents the statistics of these unified categories.

Baselines. We compare our approach against the following baselines: (1) **Full Context**, which inputs the complete conversation history directly; (2) **Naive RAG** (Gao et al., 2023), evaluated with both *session-level* and *turn-level* retrieval granularities; (3) **LangMem** (Chase, 2022), which extracts salient facts for vector-based retrieval; (4) **A-Mem** (Xu et al., 2025), which builds knowledge graphs with dynamic note linking; and (5) **Mem0** (Chhikara et al., 2025), which utilizes incremental fact compression with graph extensions.

Metrics. We evaluate performance based on both effectiveness and efficiency. For effectiveness, we report **Accuracy (ACC)**, utilizing LLM-as-LLM-Judge to determine if the response correctly answers the question. Unlike surface-level metrics such as F1 or BLEU which rely on lexical overlap, this approach better captures semantic alignment. For efficiency, we report Token Consumption and Total Runtime as end-to-end metrics, encompassing memory construction, retrieval, and LLM generation. We include these metrics because many existing memory architectures prioritize performance over efficiency, often overlooking the significant computational overhead and latency associated with building and maintaining complex memory structures. Such neglect limits their practical deployment in real-world scenarios.

Implementation Details. We employ GPT-4o-mini (Hurst et al., 2024) and Qwen3-32B (Non-Thinking Mode) (Team, 2025) as the backbone Large Language Models (LLMs) for both answer generation and the LLM-as-Judge evaluation.For semantic representation, we utilize Qwen3-Embedding-8B (Zhang et al., 2025b) to encode memory nodes and queries. Regarding retrieval, the number of retrieved segments (k) is set to 20 across all datasets. To ensure a fair comparison, we standardize the answer generation prompts for all experiments. Detailed prompts for generation and judge are provided in Appendix A.1

4.2 Main Results

Table 2 presents the comparative performance of CoM against various baselines on the LongMemEval and LoCoMo benchmarks. CoM demonstrates a superior trade-off between effectiveness and efficiency, achieving state-of-the-art accuracy while maintaining a minimal computational cost.

Method	LongMemEval							LoCoMo						
	Accuracy (%) ↑					Cost ↓		Accuracy (%) ↑					Cost ↓	
	S-hop	M-hop	Temp	Kno	Total	Token	Time	S-hop	M-hop	Temp	Kno	Total	Token	Time
Backbone: GPT-4o-mini														
Full-Context	68.59	39.85	44.36	76.92	55.80	112.7	8154	**88.59**	**53.55**	50.78	50.00	71.88	31.8	4933
RAG (session)	75.64	54.89	44.36	57.69	59.00	26.6	1512	78.83	52.02	47.16	44.79	65.32	7.5	2601
RAG (turn)	79.49	54.14	48.87	76.92	64.20	**7.6**	**1293**	77.76	41.13	44.79	**60.44**	65.39	**1.4**	**2304**
LangMem	49.35	38.35	33.83	35.89	40.20	124.7	8914	66.46	36.17	41.74	38.54	54.02	286.8	3318
A-Mem	73.72	42.86	34.59	55.13	52.20	289.2	41768	74.31	37.58	44.54	37.50	59.09	357.0	8787
Mem0	41.67	45.11	54.14	64.10	49.40	149.5	28377	73.72	40.07	61.05	47.91	63.31	388.4	6326
CoM	**84.62**	**65.41**	**65.41**	**83.33**	**74.20**	8.2	2730	83.59	48.94	**73.52**	46.88	**72.86**	4.4	3058
Backbone: Qwen3-32B														
Full-Context	73.72	41.35	38.35	70.51	55.20	119.6	11504	**86.44**	**51.06**	49.53	45.83	69.74	36.0	2981
RAG (session)	78.21	54.89	47.37	66.67	62.00	27.3	**1598**	78.83	45.39	44.85	39.58	63.18	7.8	**1816**
RAG (turn)	82.05	59.40	48.87	74.36	66.00	**7.9**	2891	75.26	34.04	57.32	41.66	61.88	**1.7**	2604
LangMem	51.28	39.10	35.33	37.17	41.60	150.2	7635	67.89	38.65	42.05	37.50	55.25	336.6	3850
A-Mem	78.20	39.10	30.08	66.67	53.20	331.8	32949	77.88	41.48	40.49	36.45	60.84	409.7	9721
Mem0	54.49	45.86	43.61	62.82	50.60	156.0	19721	75.74	45.74	53.27	42.70	63.51	456.0	8739
CoM	**86.54**	**69.92**	**69.17**	**79.49**	**76.40**	8.8	2002	81.45	46.09	**71.96**	**48.95**	**70.97**	4.8	2050

Table 2: Main results on the LongMemEval and LoCoMo benchmarks. We report Accuracy (%) across four sub-tasks: **S-hop** (Single-hop), **M-hop** (Multi-hop), **Temp** (Temporal), and **Kno** (Knowledge). **Total** denotes the accuracy calculated across the entire dataset (all samples). **Token** refers to the total end-to-end token consumption (in thousands, k), and **Time** indicates the total runtime in seconds (s). Best results are highlighted in bold.

Performance and Effectiveness. Our method demonstrates significant improvements over existing state-of-the-art baselines. On the **LongMemEval** benchmark, CoM secures the highest total accuracy. With the Qwen3-32B backbone, it achieves **76.40%**, exceeding the strongest baseline by approximately **15.8%**. These gains are particularly pronounced in logic-intensive sub-tasks, such as *Multi-hop* and *Temporal Reasoning*, where our method effectively bridges information gaps that limit other approaches. Similarly, on the **LoCoMo** benchmark, CoM continues to lead, outperforming the best baseline by over **14%** in total accuracy (70.97% vs. 61.88% for Qwen3-32B). Notably, in the challenging *Temporal* task, our method shows a remarkable advantage, improving accuracy by nearly **25%** compared to the closest competitor. These consistent improvements show our dynamic chaining mechanism successfully organizes isolated fragments into coherent inference paths, addressing the "isolated memory" limitation observed in prior works.

Efficiency and Cost-Benefit Analysis. A critical observation from Table 2 is the inefficiency of existing complex memory organizations. Despite their intricate designs, methods like A-Mem and Mem0 perform comparably to—or sometimes worse than—simple baselines, failing to justify their complexity. More strikingly, these structures incur prohibitive computational costs that can exceed even the *Full-Context* baseline. For example, constructing and querying A-Mem with Qwen3 requires nearly **332k** tokens—almost **3**$\times$ the consumption of processing the entire full context (119.56k)—yet it yields lower accuracy on LongMemEval. In sharp contrast, CoM maintains a highly efficient profile. It operates with a minimal token footprint (approx. **8.8k**), which is comparable to the most lightweight baselines. Furthermore, regarding latency, CoM significantly reduces the total runtime compared to graph-based methods (e.g., **2,002s** vs. 32,949s for A-Mem), avoiding the overhead of complex structure traversal.

Comparison with re-ranking. We further compare CoM with a cross-encoder re-ranking baseline in Appendix A.2. The results show that stronger post-retrieval scoring alone cannot match CoM, suggesting that context-aware chain organization is essential for the observed gains.

Gating Variant	LongMemEval	LoCoMo
Only Global Relevance	66.15	65.26
Only Contextual Consistency	69.35	66.84
Weighted Average	72.55	68.17
Multiplicative Combination	**76.40**	**70.97**

Table 3: Ablation of the state-aware gating formulation using Qwen3-32B. Accuracy is reported in percentage.

Dataset	$\beta = 0.3$	$\beta = 0.5$	$\beta = 0.7$
LongMemEval	73.95	**76.40**	70.14
LoCoMo	67.52	**70.97**	66.48

Table 4: Analysis of the adaptive truncation threshold β using Qwen3-32B. Accuracy is reported in percentage.

4.3 Ablation Study

Effect of framework components. To rigorously assess the contribution of each component, Figure 3 details our ablation study on Dynamic Memory Chain Evolution (DMCE) and Adaptive Path Truncation (APT). First, removing the entire framework causes accuracy on GPT-4o-mini to plummet to 55.80%, accompanied by a massive $13\times$ surge in token consumption (112.66 vs. 8.24). This confirms that our architecture is foundational for maintaining long-context coherence while minimizing redundancy. Second, excluding DMCE leads to a significant degradation, with accuracy falling to 59.00%. Although the *w/o DMCE* variant yields marginally lower latency on Qwen3-32B, it suffers a severe drop in answer quality, proving that Chain Evolution is critical for sustaining information density without imposing heavy overhead. Finally, omitting the APT stage results in suboptimal accuracy (72.75%) and increased computational costs. This outcome underscores the necessity of Path Truncation in filtering out irrelevant noise and balancing efficiency with precision.

Effect of the gating formulation. We further analyze the state-aware gating score by replacing the multiplicative formulation with three alternatives: using only global relevance, using only contextual consistency, and using a weighted average of the two scores. As shown in Table 3, the multiplicative formulation performs best on both datasets. This supports our design choice of treating query relevance and contextual consistency as a soft conjunction: a useful successor should be relevant to the original query while also coherent with the evolving reasoning chain.

4.4 Hyperparameter Analysis

Effect of retrieval size. We investigate the impact of the retrieval hyperparameter k ($k \in \{1, 5, 10, 20, 50\}$) on LongMemEval using the Qwen3-32B backbone. As illustrated in Figure 4, the performance gains from increasing k eventually saturate, suggesting that pivotal information is predominantly concentrated within the top retrieved chunks. Consequently, excessively increasing k yields diminishing returns, as the inclusion of irrelevant context introduces noise that may hamper reasoning. In terms of efficiency, although token consumption grows linearly with k, total runtime remains largely stable because latency is dominated by LLM inference and network overhead rather than retrieval. Thus, a moderate k provides an effective trade-off between context sufficiency and computational efficiency.

Sensitivity to the truncation threshold. We evaluate the adaptive truncation threshold β, which controls when chain expansion stops after a sharp score drop. Table 4 shows that $\beta = 0.5$ performs best on both LongMemEval and LoCoMo. A smaller threshold tends to under-truncate and introduce noisy evidence, while a larger threshold may stop expansion too early and miss necessary intermediate evidence. The consistent trend across two benchmarks suggests that CoM does not require dataset-specific tuning of β.

Robustness across embedding models. To examine whether CoM depends on a particular embedding model, we replace Qwen3-Embedding-8B with `text-embedding-3-small`, keeping Qwen3-32B as the backbone LLM. As shown in Appendix A.3, CoM consistently outperforms RAG on both LongMemEval and LoCoMo. This suggests that CoM's gains do not rely on a specific embedding backbone, but stem from organizing retrieved fragments into context-aware memory chains.

4.5 Upper Bound Analysis

To rigorously benchmark the theoretical limits and efficacy of our approach, we establish two distinct baselines: a state-of-the-art long-context model and an oracle setting for our backbone models.

Long-Context Baseline. We employ *Gemini-2.5-pro*, utilizing its 1M token context window to process the full input without truncation. Given that *LongMemEval* samples average 105k tokens, this

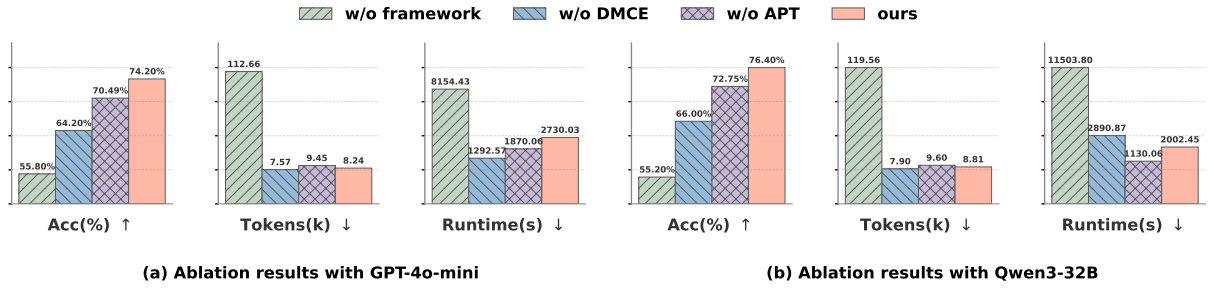

Figure 3: **Ablation Study Results.** We compare the performance of our full method against variants removing specific components (w/o Framework, w/o DMCE, w/o APT) on GPT-4o-mini (a) and Qwen3-32B (b). The metrics include Accuracy (Acc), Token consumption, and Runtime. Our method achieves the best trade-off between accuracy and efficiency.

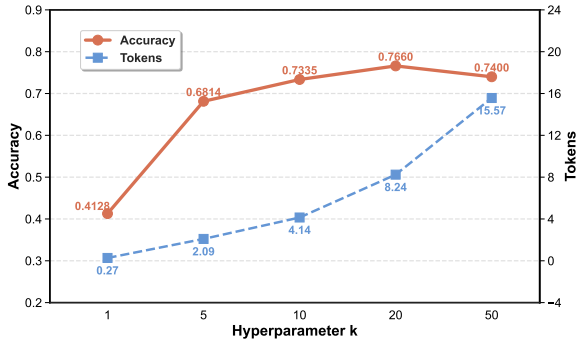

Figure 4: The impact of hyperparameter k on model accuracy and computational cost (tokens).

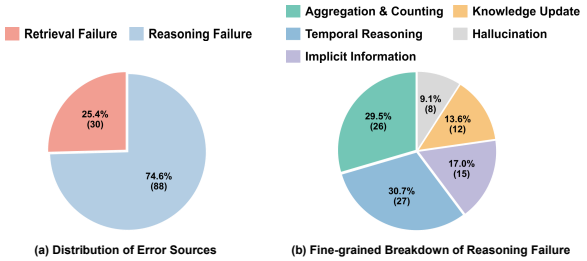

Figure 5: Distribution of error types on the LongMemEval dataset. Reasoning failures constitute the majority of errors compared to retrieval failure.

baseline effectively bypasses retrieval-induced information loss, serving as a robust upper bound for lossless context understanding.

Oracle Setting (Evidence Only). To isolate the intrinsic reasoning capabilities of our backbones (*GPT-4o-mini* and *Qwen3-32B*) from retrieval errors, we evaluate an Oracle setting. Here, models are fed only the manually annotated ground-truth evidence. This configuration eliminates noise, quantifying the performance ceiling achievable when perfect information is guaranteed.

Analysis and Conclusion. As detailed in Table 5, *Gemini-2.5-pro* achieves the highest accuracy (Total: 89.20%), confirming the advantage of full-context processing. However, this precision entails steep computational costs (122.89k tokens, 8512.10s). The Oracle results demarcate the reasoning limits of the smaller backbones; our method approaches this ceiling (e.g., 76.40% vs. 81.80% for Qwen3), indicating that our approach successfully captures the majority of critical information. Crucially, while *Gemini-2.5-pro* offers marginal accuracy gains, our method is significantly more efficient—reducing token consumption by $14\times$ and latency by $4\times$. This presents a superior trade-off for practical applications where the overhead of processing massive contexts is prohibitive.

4.6 Error Analysis

To systematically diagnose failure modes, we conducted a comprehensive analysis of 118 error samples from the LongMemEval dataset. As illustrated in Figure 5(a), reasoning failures constitute the vast majority of errors (74.6%), whereas retrieval failures account for a relatively minor portion (25.4%). This distribution suggests that the primary performance bottleneck lies in the model's capability to interpret and reason over context, rather than the inability to access relevant information.

Retrieval Failure This category accounts for a minority of errors. Primary causes for retrieval failure include information dispersion in multi-hop questions, middle-position information loss in long contexts, and the semantic misalignment often encountered between queries and source documents.

Reasoning Failure Reasoning errors dominate the failure modes, persisting even after successful evidence retrieval. In these cases, the model ac-

Method	Sub-category Acc (%)				Total Acc (%)	Tokens (k)	RunTime (s)
	Single-hop	Multi-hop	Temporal	Knowledge			
Full Context (Gemini-2.5-pro)	**90.38**	**86.47**	**90.98**	**88.46**	**89.20**	122.89	8512.10
Evidence Only (GPT-4o-mini)	83.33	82.71	69.17	**88.46**	80.20	**0.30**	668.36
Ours (GPT-4o-mini)	84.62	65.41	65.41	83.33	74.20	8.24	2730.03
Evidence Only (Qwen3-32B)	89.74	81.20	70.68	85.90	81.80	0.35	**456.20**
Ours (Qwen3-32B)	86.54	69.92	69.17	79.49	76.40	8.81	2002.45

Table 5: Performance comparison between Full-Context SOTA (Gemini-2.5-pro), Evidence-Only (with only the ground-truth target information required to answer the question), and our method on LongMemEval dataset.

cesses the context but fails to effectively extract or interpret the information. We categorize these failures into five subtypes: (1) **Temporal Reasoning (30.7%):** Models struggle to reconstruct accurate chronological timelines, frequently misinterpreting relative expressions or misordering sequential events; (2) **Aggregation & Counting (29.5%):** Prominent in questions requiring statistical synthesis, this error arises when models fail to distinguish between entities actively *acquired* and those merely *mentioned*, resulting in systematic overcounting; (3) **Implicit Information (17.0%):** Models lack the sensitivity to infer latent user intents or constraints from behavioral patterns, consequently generating generic responses that lack necessary personalization; (4) **Knowledge Update (13.6%):** Models exhibit inertia in tracking dynamic state changes, often prioritizing semantically relevant but outdated information over the most recent evidence; and (5) **Hallucination (9.1%):** Occurring primarily in abstention tasks, models fabricate details based on tangential associations instead of correctly acknowledging the lack of evidence.

5 Conclusion

This work shows that effective agent memory can be achieved without increasingly complex memory construction. We propose CoM, a Chain-of-Memory framework that shifts the focus to lightweight retrieval and principled memory utilization by organizing retrieved fragments into dynamic inference paths and pruning distracting contexts through adaptive truncation. Experiments on LongMemEval and LoCoMo show that CoM improves reasoning accuracy, especially on multi-hop and temporal questions, while substantially reducing token consumption and runtime. These results demonstrate that structured memory utilization can bridge the gap between retrieved evidence and final reasoning accuracy. Rather than relying on heavier memory structures, CoM improves how retrieved fragments are connected, filtered, and presented to the LLM. Overall, our findings highlight lightweight construction, context-aware memory organization, and system-level efficiency as practical directions for building efficient long-term memory systems for LLM agents.

Limitations

Despite its effectiveness, CoM still has several limitations. First, CoM relies on embedding-based semantic similarity to retrieve and evolve memory chains. Its performance may therefore degrade when the embedding model fails to capture fine-grained temporal, pragmatic, or implicit relations between memory fragments. Second, although adaptive truncation helps reduce noise and redundancy, it may discard subtle evidence in cases that require unusually long or weakly connected reasoning paths. Third, our experiments focus on textual long-term memory benchmarks, leaving multi-modal memories, such as vision-language interaction histories and tool-use traces, for future work. Finally, real-world agents may involve more open-ended goals, evolving user preferences, and noisier memory updates than current benchmarks cover. Further validation in interactive deployment settings is needed to assess the practical robustness of CoM under such conditions.

Acknowledgments

We would like to thank the Director's Fund Project of State Key Laboratory of AI Safety for their support. This work was also supported in part by the Strategic Priority Research Program of the CAS (No. XDB0680302) and the Young Elite Scientists Sponsorship Program of the Beijing High Innovation Plan (No. 20250924).

References

Harrison Chase. 2022. LangChain.

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

Junnan Dong, Siyu An, Yifei Yu, Qian-Wen Zhang, Linhao Luo, Xiao Huang, Yunsheng Wu, Di Yin, and Xing Sun. 2025. Youtu-graphrag: Vertically unified agents for graph retrieval-augmented complex reasoning. *arXiv preprint arXiv:2508.19855*.

André V Duarte, João DS Marques, Miguel Graça, Miguel Freire, Lei Li, and Arlindo L Oliveira. 2024. Lumberchunker: Long-form narrative document segmentation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 6473–6486.

Weizhi Fei, Xueyan Niu, Pingyi Zhou, Lu Hou, Bo Bai, Lei Deng, and Wei Han. 2024. Extending context window of large language models via semantic compression. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 5169–5181.

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

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2024. Lightrag: Simple and fast retrieval-augmented generation. *arXiv preprint arXiv:2410.05779*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Zhiyu Li, Shichao Song, Hanyu Wang, Simin Niu, Ding Chen, Jiawei Yang, Chenyang Xi, Huayi Lai, Jihao Zhao, Yezhaohui Wang, and 1 others. 2025. Memos: An operating system for memory-augmented generation (mag) in large language models. *arXiv preprint arXiv:2505.22101*.

Bang Liu, Xinfeng Li, Jiayi Zhang, Jinlin Wang, Tanjin He, Sirui Hong, Hongzhang Liu, Shaokun Zhang, Kaitao Song, Kunlun Zhu, and 1 others. 2025a. Advances and challenges in foundation agents: From brain-inspired intelligence to evolutionary, collaborative, and safe systems. *arXiv preprint arXiv:2504.01990*.

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.

Zuhong Liu, Charles-Elie Simon, and Fabien Caspani. 2025b. Passage segmentation of documents for extractive question answering. In *European Conference on Information Retrieval*, pages 345–352. Springer.

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

Lingrui Mei, Jiayu Yao, Yuyao Ge, Yiwei Wang, Baolong Bi, Yujun Cai, Jiazhi Liu, Mingyu Li, Zhong-Zhi Li, Duzhen Zhang, and 1 others. 2025. A survey of context engineering for large language models. *arXiv preprint arXiv:2507.13334*.

Charles Packer, Vivian Fang, Shishir_G Patil, Kevin Lin, Sarah Wooders, and Joseph_E Gonzalez. 2023. Memgpt: Towards llms as operating systems.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Xufang Luo, Hao Cheng, Dongsheng Li, Yuqing Yang, Chin-Yew Lin, H Vicky Zhao, Lili Qiu, and 1 others. 2025. Secom: On memory construction and retrieval for personalized conversational agents. In *The Thirteenth International Conference on Learning Representations*.

Renyi Qu, Ruixuan Tu, and Forrest Bao. 2025. Is semantic chunking worth the computational cost? In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 2155–2177.

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

Alireza Rezazadeh, Zichao Li, Wei Wei, and Yujia Bao. 2024. From isolated conversations to hierarchical schemas: Dynamic tree memory representation for llms. *arXiv preprint arXiv:2410.14052*.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D Manning. 2024. Raptor: Recursive abstractive processing for tree-organized retrieval. In *The Twelfth International Conference on Learning Representations*.

Joykirat Singh, Raghav Magazine, Yash Pandya, and Akshay Nambi. 2025. Agentic reasoning and tool integration for llms via reinforcement learning. *arXiv preprint arXiv:2505.01441*.

Zhongxiang Sun, Qipeng Wang, Weijie Yu, Xiaoxue Zang, Kai Zheng, Jun Xu, Xiao Zhang, Yang Song, and Han Li. 2025. Rearter: Retrieval-augmented reasoning with trustworthy process rewarding. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1251–1261.

Qwen Team. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Bing Wang, Xinnian Liang, Jian Yang, Hui Huang, Shuangzhi Wu, Peihao Wu, Lu Lu, Zejun Ma, and Zhoujun Li. 2023. Enhancing large language model with self-controlled memory framework. *arXiv preprint arXiv:2304.13343*.

Cangqing Wang, Yutian Yang, Ruisi Li, Dan Sun, Ruicong Cai, Yuzhu Zhang, and Chengqian Fu. 2024. Adapting llms for efficient context processing through soft prompt compression. In *Proceedings of the International Conference on Modeling, Natural Language Processing and Machine Learning*, pages 91–97.

Zora Zhiruo Wang, Apurva Gandhi, Graham Neubig, and Daniel Fried. 2025. Inducing programmatic skills for agentic tasks. *arXiv preprint arXiv:2504.06821*.

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

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

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652.

Wenhao Yu, Hongming Zhang, Xiaoman Pan, Peixin Cao, Kaixin Ma, Jian Li, Hongwei Wang, and Dong Yu. 2024. Chain-of-note: Enhancing robustness in retrieval-augmented language models. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pages 14672–14685.

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, Yijiang Li, and 1 others. 2025a. The landscape of agentic reinforcement learning for llms: A survey. *arXiv preprint arXiv:2509.02547*.

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. 2025b. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

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

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

Jihao Zhao, Zhiyuan Ji, Yuchen Feng, Pengnian Qi, Simin Niu, Bo Tang, Feiyu Xiong, and Zhiyu Li. 2024b. Meta-chunking: Learning text segmentation and semantic completion via logical perception. *arXiv preprint arXiv:2410.12788*.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 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 Appendix

A.1 Prompts

LongMemEval Answer Prompt

System Prompt:
You are a helpful expert assistant answering questions from the user based on the provided context.

User Prompt:
Your task is to briefly answer the question. You are given the following context from the previous conversation. If you don't know how to answer the question, abstain from answering.

Context: `{context}`
Question: `{question}`

LoCoMo Answer Prompt

System Prompt:
You are a **Memory-Based Question Answering Agent**. Answer questions based on the provided memories.

<Instructions>

1. **For time-related questions (with words like "when"):**
   - **Step 1:** Identify the Timestamp of the memory.
   - **Step 2:** Identify any relative time reference.
   - **Step 3:** Calculate the **EXACT ABSOLUTE** date.
   - *Example 1:* [Timestamp: 4 May 2022] + "last year" → Answer: "last year before 4 May 2022"
2. **Extract specific details:** Use exact names, dates, places from memories.
3. **Be comprehensive for list questions:** List ALL activities found.
4. **Contradictions:** If memories conflict, use the most recent one.

</Instructions>

User Prompt:

```
<Memories>
{memories}
</Memories>

<Question>{question}</Question>

Answer:
```

LongMemEval Judge Prompt

You are an expert judge evaluating whether a model's response correctly answers a question according to the reference answer or rubric.

1. Basic Evaluation
I will give you a question, a correct answer, and a response from a model. Please answer yes if the response contains the correct answer.

Question: `{}`
Correct Answer: `{}`
Model Response: `{}`

Is the model response correct? Answer yes or no only.

---

2. Temporal Reasoning
I will give you a question, a correct answer, and a response from a model. If the response is equivalent or contains all intermediate steps, answer yes. Do not penalize off-by-one errors for the number of days.

Question: `{}`
Correct Answer: `{}`
Model Response: `{}`

Is the model response correct? Answer yes or no only.

3. Knowledge Update
If the response contains some previous information along with an updated answer, the response should be considered correct.

Question: {}
Correct Answer: {}
Model Response: {}

Is the model response correct? Answer yes or no only.

---

4. User Preference
Please answer yes if the response satisfies the desired response based on the rubric.

Question: {}
Rubric: {}
Model Response: {}

Is the model response correct? Answer yes or no only.

---

5. Abstention (Unanswerable)
Please answer yes if the model correctly identifies the question as unanswerable.

Question: {}
Explanation: {}
Model Response: {}

Does the model correctly identify the question as unanswerable? Answer yes or no only.

LoCoMo Judge Prompt

You are an expert judge evaluating whether a model's prediction correctly answers a question compared to the reference answer.

Question: {}
Reference Answer: {}
Model Prediction: {}

Your task is to determine if the model's prediction is semantically equivalent to the reference answer. Consider the following:
1. The prediction may be phrased differently but convey the same meaning.
2. Minor differences in wording are acceptable if the core information matches.
3. For dates, consider different formats as equivalent (e.g., "7 May 2023" vs "May 7, 2023").
4. For numbers, consider "2022" vs "Last year" as potentially equivalent depending on context.
5. For descriptive answers, check if the key information is present.

Respond with ONLY ONE WORD:
- "CORRECT" if the prediction matches the reference answer.
- "INCORRECT" if the prediction does not match the reference answer.

Your response:

Method	Acc.	Tokens	Time
Rerank (GPT-4o-mini)	62.73	11.01	3967.52
CoM (GPT-4o-mini)	**74.20**	**8.24**	**2730.03**
Rerank (Qwen3-32B)	63.20	10.95	2702.47
CoM (Qwen3-32B)	**76.40**	**8.81**	**2002.45**

Table 6: Comparison with a cross-encoder re-ranking baseline on LongMemEval. Tokens are reported in thousands.

Method	Acc.	Tokens	Time
Rerank (GPT-4o-mini)	68.31	**3.53**	3413.43
CoM (GPT-4o-mini)	**72.86**	4.35	**3057.50**
Rerank (Qwen3-32B)	65.32	**3.69**	2805.66
CoM (Qwen3-32B)	**70.97**	4.77	**2050.48**

Table 7: Comparison with a cross-encoder re-ranking baseline on LoCoMo. Tokens are reported in thousands.

A.2 Comparison with Re-ranking and Iterative Retrieval

CoM is related to post-retrieval re-ranking and iterative retrieval, but differs in both objective and operation. Re-ranking methods typically assign each retrieved segment an independent relevance score with respect to the query, and then select a reordered top-k set. Iterative retrieval methods often expand or reformulate the retrieval query over multiple rounds to retrieve additional evidence. In contrast, CoM keeps memory construction and initial retrieval lightweight, and focuses on organizing the retrieved fragments into chain-structured inference paths. Each expansion step is conditioned not only on the original query but also on the evolving chain context, allowing CoM to recover multi-hop and temporal dependencies that may not be captured by global relevance alone.

To test whether stronger post-retrieval scoring can explain CoM's gains, we compare with a cross-encoder re-ranking baseline. This baseline first retrieves an expanded candidate pool and then uses `bge-reranker-base` to select the final top-k segments for generation. As shown in Table 6 and Table 7, even with a dedicated re-ranker, this baseline remains below CoM on both benchmarks. This suggests that CoM's improvements do not merely come from assigning better independent relevance scores, but from dynamically organizing retrieved fragments into context-aware memory chains.

A.3 Robustness Across Embedding Models

To evaluate whether CoM depends on a specific embedding model, we replace Qwen3-Embedding-8B with `text-embedding-3-small` and compare CoM with RAG using Qwen3-32B as the backbone LLM. Tables 8 and 9 show that CoM consistently improves overall accuracy on both benchmarks.

Method	S-hop	M-hop	Temp	Kno	Overall
RAG	80.77	54.14	53.38	74.36	65.40
CoM	**84.62**	**68.42**	**67.67**	**79.49**	**75.00**

Table 8: Performance on LongMemEval using `text-embedding-3-small` with Qwen3-32B.

Method	S-hop	M-hop	Temp	Kno	Overall
RAG	74.31	43.26	66.35	43.75	65.06
CoM	**81.92**	**43.61**	**69.47**	**46.87**	**70.12**

Table 9: Performance on LoCoMo using `text-embedding-3-small` with Qwen3-32B.