

SimpleMem: Efficient Lifelong Memory for LLM Agents

Jiaqi Liu^{1*} Yaofeng Su^{1*} Peng Xia¹ Siwei Han¹
 Zeyu Zheng² Cihang Xie³ Mingyu Ding¹ Huaxiu Yao¹

Abstract

To support reliable long-term interaction in complex environments, LLM agents require memory systems that efficiently manage historical experiences. Existing approaches either retain full interaction histories via passive context extension, leading to substantial redundancy, or rely on iterative reasoning to filter noise, incurring high token costs. To address this challenge, we introduce SimpleMem, an efficient memory framework based on semantic lossless compression. We propose a three-stage pipeline designed to maximize information density and token utilization: (1) *Semantic Structured Compression*, which applies entropy-aware filtering to distill unstructured interactions into compact, multi-view indexed memory units; (2) *Recursive Memory Consolidation*, an asynchronous process that integrates related units into higher-level abstract representations to reduce redundancy; and (3) *Adaptive Query-Aware Retrieval*, which dynamically adjusts retrieval scope based on query complexity to construct precise context efficiently. Experiments on benchmark datasets show that our method consistently outperforms baseline approaches in accuracy, retrieval efficiency, and inference cost, achieving an average F1 improvement of 26.4% while reducing inference-time token consumption by up to 30x, demonstrating a superior balance between performance and efficiency. Code is available at <https://github.com/aiming-lab/SimpleMem>.

1. Introduction

Large Language Model (LLM) agents have recently demonstrated remarkable capabilities across a wide range of

*Equal contribution ¹UNC-Chapel Hill ²University of California, Berkeley ³University of California, Santa Cruz. Correspondence to: Jiaqi Liu <jqliu@cs.unc.edu>, Mingyu Ding <md@cs.unc.edu>, Huaxiu Yao <huaxiu@cs.unc.edu>.

Preprint. January 7, 2026.

tasks (Xia et al., 2025; Team et al., 2025; Qiu et al., 2025). However, constrained by fixed context windows, existing agents exhibit significant limitations when engaging in long-context and multi-turn interaction scenarios (Liu et al., 2023; Wang et al., 2024a; Liu et al., 2025; Hu et al., 2025; Tu et al., 2025). To facilitate reliable long-term interaction, LLM agents require robust memory systems to efficiently manage and utilize historical experience (Dev & Taranjeet, 2024; Fang et al., 2025; Wang & Chen, 2025; Tang et al., 2025; Yang et al., 2025; Ouyang et al., 2025).

While recent research has extensively explored the design of memory modules for LLM agents, current systems still suffer from suboptimal retrieval efficiency and low token utilization (Fang et al., 2025; Hu et al., 2025). On one hand, many existing systems maintain complete interaction histories through full-context extension (Li et al., 2025; Zhong et al., 2024). However, this approach introduce substantial redundant information (Hu et al., 2025). Specifically, during long-horizon interactions, user inputs and model responses accumulate substantial low-entropy noise (e.g., repetitive logs, non-task-oriented dialogue), which degrades the effective information density of the memory buffer. This redundancy adversely affects memory retrieval and downstream reasoning, often leading to middle-context degradation phenomena (Liu et al., 2023), while also incurring significant computational overhead during retrieval and secondary inference. On the other hand, some agentic frameworks mitigate noise through online filtering based on iterative reasoning procedures (Yan et al., 2025; Packer et al., 2023). Although such approaches improve retrieval relevance, they rely on repeated inference cycles, resulting in substantial computational cost, including increased latency and token usage. As a result, neither paradigm achieves efficient allocation of memory and computation resources.

To address these limitations, we introduce **SimpleMem**, an efficient memory framework inspired by the Complementary Learning Systems (CLS) theory (Kumaran et al., 2016) and designed around structured semantic compression. The core objective of SimpleMem is to improve information efficiency under fixed context and token budgets. To this end, we develop a three-stage pipeline that supports dynamic memory compression, organization, and adaptive retrieval: (1) **Semantic Structured Compression**: we ap-

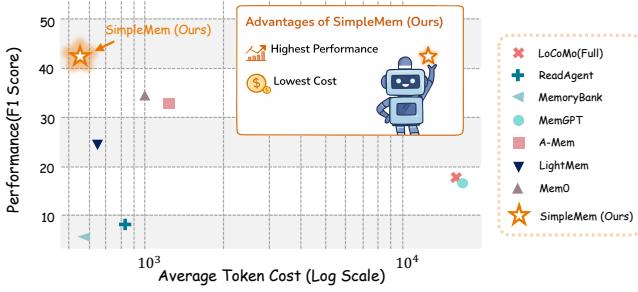


Figure 1. Performance vs. Efficiency Trade-off. Comparison of Average F1 against Average Token Cost on the LoCoMo benchmark. SimpleMem occupies the ideal top-left position, achieving high accuracy with minimal token consumption (~ 550 tokens).

ply an entropy-aware filtering mechanism that preserves information with high semantic utility while discarding redundant or low-value content. The retained information is reformulated into compact memory units and jointly indexed using dense semantic embeddings, sparse lexical features, and symbolic metadata, enabling multi-granular retrieval. (2) **Recursive Memory Consolidation:** Inspired by biological consolidation, we introduce an asynchronous process that incrementally reorganizes stored memory. Rather than accumulating episodic records verbatim, related memory units are recursively integrated into higher-level abstract representations, allowing repetitive or structurally similar experiences to be summarized while reducing semantic redundancy. (3) **Adaptive Query-Aware Retrieval:** we employ a query-aware retrieval strategy that dynamically adjusts retrieval scope based on estimated query complexity. Irrelevant candidates are pruned through lightweight symbolic and semantic constraints, enabling precise context construction tailored to task requirements. This adaptive mechanism achieves a favorable trade-off between reasoning performance and token efficiency.

Our primary contribution is SimpleMem, an efficient memory framework grounded in structured semantic compression, which improves information efficiency through principled memory organization, consolidation, and adaptive retrieval. As shown in Figure 1, our empirical experiments demonstrate that SimpleMem establishes a new state-of-the-art with an F1 score, outperforming strong baselines like Memo by 26.4%, while reducing inference token consumption by 30 \times compared to full-context models.

2. The SimpleMem Architecture

In this section, we present **SimpleMem**, an efficient memory framework for LLM agents designed to improve information utilization under constrained context and token budgets through. As shown in Figure 2, the system operates through a three-stage pipeline. First, we describe *Semantic Structured Compression* process, which filters redundant interaction content and reformulates raw dialogue

streams into compact memory units. Next, we describe *Recursive Consolidation*, an asynchronous process that incrementally integrates related memory units into higher-level abstract representations and maintaining a compact memory topology. Finally, we present *Adaptive Query-Aware Retrieval*, which dynamically adjusts retrieval scope based on estimated query complexity to construct precise and token-efficient contexts for downstream reasoning.

2.1. Semantic Structured Compression

A primary bottleneck in long-term interaction is *context inflation*, the accumulation of raw, low-entropy dialogue. For example, a large portion of interaction segments in the real-world consists of phatic chit-chat or redundant confirmations, which contribute little to downstream reasoning but consume substantial context capacity. To address this, we introduce a mechanism to actively filter and restructure information at the source.

First, incoming dialogue is segmented into overlapping sliding windows W_t of fixed length, where each window represents a short contiguous span of recent interaction. These windows serve as the basic units for evaluating whether new information should be stored. Then we employ a non-linear gating mechanism, Φ_{gate} , to evaluate the information density of these dialogue windows to determine which windows is used fo indexing. For each window W_t , we compute an information score $H(W_t)$ that jointly captures the introduction of new entities and semantic novelty relative to the immediate interaction history H_{prev} .

Formally, let \mathcal{E}_{new} denote the set of named entities that appear in W_t but not in H_{prev} . The information score is defined as:

$$H(W_t) = \alpha \cdot \frac{|\mathcal{E}_{new}|}{|W_t|} + (1-\alpha) \cdot (1 - \cos(E(W_t), E(H_{prev}))) \quad (1)$$

where $E(\cdot)$ denotes a semantic embedding function and α controls the relative importance of entity-level novelty and semantic divergence.

Windows whose information score falls below threshold $\tau_{redundant}$ are treated as redundant and excluded from memory construction, meaning that the window is neither stored nor further processed, preventing low-utility interaction content from entering the memory buffer. For informative windows, the system proceeds to a segmentation step:

$$\text{Action}(W_t) = \begin{cases} \text{Segment}(W_t), & H(W_t) \geq \tau_{redundant}, \\ \emptyset, & \text{otherwise.} \end{cases} \quad (2)$$

For windows that pass the filter, we apply a segmentation function \mathcal{F}_θ to decompose each informative window into a set of context-independent memory units m_k . This trans-

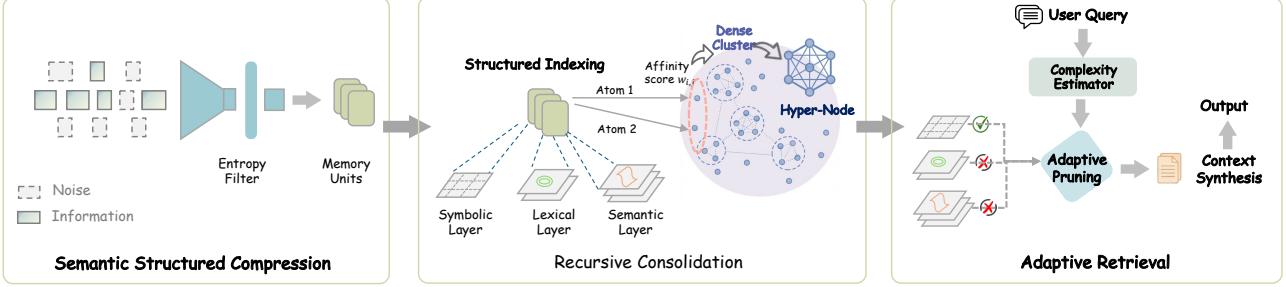


Figure 2. The SimpleMem Architecture. SimpleMem mitigates context inflation through three stages. (1) Semantic Structured Compression filters redundant interaction content and reformulates raw dialogue into compact, context-independent memory units. (2) Recursive Consolidation incrementally organizes related memory units into higher-level abstract representations, reducing redundancy in long-term memory. (3) Adaptive Query-Aware Retrieval dynamically adjusts retrieval scope based on query complexity, enabling efficient context construction under constrained token budgets.

formation resolves dependencies implicit in conversational flow by converting entangled dialogue into self-contained factual or event-level statements. Formally, $\mathcal{F}\theta$ is composed of a coreference resolution module (Φ_{coref}) and a temporal anchoring module (Φ_{time}):

$$m_k = \mathcal{F}_\theta(W_t) = \Phi_{time} \circ \Phi_{coref} \circ \Phi_{extract}(W_t) \quad (3)$$

Here, $\Phi_{extract}$ identifies candidate factual statements, (Φ_{coref}) replaces ambiguous pronouns with specific entity names (e.g., changing "He agreed" to "Bob agreed"), and Φ_{time} converts relative temporal expressions (e.g., transforming "next Friday" to "2025-10-24") into absolute ISO-8601 timestamps. This normalization ensures that each memory unit remains interpretable and valid independent of its original conversational context.

2.2. Structured Indexing and Recursive Consolidation

Then, the system need organize the resulting memory units to support efficient long-term storage and scalable retrieval. This stage consists of two components: (i) structured multi-view indexing for immediate access, and (ii) recursive consolidation for reducing redundancy and maintaining a compact memory topology over time.

To support flexible and precise retrieval, each memory unit is indexed through three complementary representations. First, at semantic layer, we map the entry to a dense vector space \mathbf{v}_k using embedding models, which captures abstract meaning and enables fuzzy matching (e.g., retrieving "latte" when querying "hot drink"). Second, the Lexical Layer generates a sparse representation focusing on exact keyword matches and proper nouns, ensuring that specific entities are not diluted in vector space. Third, the Symbolic Layer extracts structured metadata, such as timestamps and entity types, to enable deterministic filtering logic. Formally, these

projections form the comprehensive memory bank \mathbb{M} :

$$\mathbb{M}(m_k) = \begin{cases} \mathbf{v}_k = E_{dense}(S_k) \in \mathbb{R}^d & (\text{Semantic Layer}) \\ \mathbf{h}_k = \text{Sparse}(S_k) \in \mathbb{R}^{|V|} & (\text{Lexical Layer}) \\ \mathcal{R}_k = \{(key, val)\} & (\text{Symbolic Layer}) \end{cases} \quad (4)$$

It allows the system to flexibly query information based on conceptual similarity, exact keyword matches, or structured metadata constraints.

While multi-view indexing supports efficient access, naively accumulating memory units over long interaction horizons leads to redundancy and fragmentation. To address this issue, we then introduce an asynchronous background consolidation process that incrementally reorganizes the memory topology. The consolidation mechanism identifies related memory units based on both semantic similarity and temporal proximity. For two memory units m_i and m_j , we define an affinity score ω_{ij} as:

$$\omega_{ij} = \beta \cdot \cos(\mathbf{v}_i, \mathbf{v}_j) + (1 - \beta) \cdot e^{-\lambda|t_i - t_j|}, \quad (5)$$

where the first term captures semantic relatedness and the second term biases the model toward grouping events with strong temporal proximity.

When a group of memory units forms a dense cluster \mathcal{C} , determined by pairwise affinities exceeding a threshold $\tau_{cluster}$, the system performs a consolidation step:

$$M_{abs} = \mathcal{G}_{syn}(\{m_i \mid m_i \in \mathcal{C}\}). \quad (6)$$

This operation synthesizes repetitive or closely related memory units into a higher-level abstract representation M_{abs} , which captures their shared semantic structure. For example, instead of maintaining numerous individual records such as "the user ordered a latte at 8:00 AM," the system consolidates them into a single abstract pattern, e.g., "the user regularly drinks coffee in the morning." The original fine-grained entries are archived, reducing the active memory size while preserving the ability to recover detailed

information if needed. As a result, the active memory index remains compact, and retrieval complexity scales gracefully with long-term interaction history.

2.3. Adaptive Query-Aware Retrieval

After memory entries are organized, another challenge to retrieve relevant information efficiently under constrained context budgets. Standard retrieval approaches typically fetch a fixed number of context entries, which often results in either insufficient information or token wastage. To address this, we introduce an adaptive query-aware retrieval mechanism that dynamically adjusts retrieval scope based on estimated query complexity, thereby improving retrieval efficiency without sacrificing reasoning accuracy.

First, we propose a hybrid scoring function for information retrieval, $\mathcal{S}(q, m_k)$, which aggregates signals from the tri-layer index established in the second stage. For a given query q , the relevance score is computed as:

$$\begin{aligned} \mathcal{S}(q, m_k) = & \lambda_1 \cos(\mathbf{e}_q, \mathbf{v}_k) + \lambda_2 \text{BM25}(q_{\text{lex}}, S_k) \\ & + \gamma \mathbb{I}(\mathcal{R}_k \models \mathcal{C}_{\text{meta}}), \end{aligned} \quad (7)$$

where the first term measures semantic similarity in the dense embedding space, the second term captures exact lexical relevance, and the indicator function $\mathbb{I}(\cdot)$ enforces hard symbolic constraints such as entity-based filters.

Then, based on the hybrid scoring, we can rank the candidate memories by relevance. However, retrieving a fixed number of top-ranked entries remains inefficient when query demands vary. To address this, we estimate the *query complexity* $C_q \in [0, 1]$, which reflects whether a query can be resolved via direct fact lookup or requires multi-step reasoning over multiple memory entries. A lightweight classifier predicts C_q based on query features such as length, syntactic structure, and abstraction level.

$$k_{\text{dyn}} = \lfloor k_{\text{base}} \cdot (1 + \delta \cdot C_q) \rfloor \quad (8)$$

Based on this dynamic depth, the system modulates the retrieval scope. For low-complexity queries ($C_q \rightarrow 0$), the system retrieves only the top- k_{\min} high-level abstract memory entries or metadata summaries, minimizing token usage. Conversely, for high-complexity queries ($C_q \rightarrow 1$), it expands the scope to top- k_{\max} , including a larger set of relevant entries, along with associated fine-grained details. The final context $\mathcal{C}_{\text{final}}$ is synthesized by concatenating these pruned results, ensuring high accuracy with minimal computational waste:

$$\mathcal{C}_{\text{final}} = \bigoplus_{m \in \text{Top-}k_{\text{dyn}}(\mathcal{S})} [t_m : \text{Content}(m)] \quad (9)$$

3. Experiments

In this section, we evaluate SimpleMem on the benchmark to answer the following research questions: (1) Does SimpleMem outperform other memory systems in complex long-term reasoning and temporal grounding tasks? (2) Can SimpleMem achieve a superior trade-off between retrieval accuracy and token consumption? (3) How effective are the proposed components? (4) What factors account for the observed performance and efficiency gains?

3.1. Experimental Setup

Benchmark Dataset. We utilize the LoCoMo benchmark (Maharana et al., 2024), which is specifically designed to test the limits of LLMs in processing long-term conversational dependencies. The dataset comprises conversation samples ranging from 200 to 400 turns, containing complex temporal shifts and interleaved topics. The evaluation set consists of 1,986 questions categorized into four distinct reasoning types: (1) Multi-Hop Reasoning: Questions requiring the synthesis of information from multiple disjoint turns (e.g., “Based on what X said last week and Y said today...”); (2) Temporal Reasoning: Questions testing the model’s ability to understand event sequencing and absolute timelines (e.g., “Did X happen before Y?”); (3) Open Domain: General knowledge questions grounded in the conversation context; (4) Single Hop: Direct retrieval tasks requiring exact matching of specific facts.

Baselines. We compare SimpleMem with representative memory-augmented systems: LOCoMo (Maharana et al., 2024), READAGENT (Lee et al., 2024), MEMORY-BANK (Zhong et al., 2024), MEMGPT (Packer et al., 2023), A-MEM (Xu et al., 2025), LIGHTMEM (Fang et al., 2025), and Mem0 (Dev & Taranjeet, 2024).

Backbone Models. To test robustness across capability scales, we instantiate each baseline and SimpleMem on multiple LLM backends: GPT-4o, GPT-4.1-mini, Qwen-Plus, Qwen2.5 (1.5B/3B), and Qwen3 (1.7B/8B).

Implementation Details. For semantic structured compression, we use a sliding window of size $W = 10$ and set the entropy-based significance threshold to $\tau = 0.35$ to filter low-information interaction content. Memory indexing is implemented using LanceDB with a multi-view design: `text-embedding-3-small` (1536 dimensions) for dense semantic embeddings, BM25 for sparse lexical indexing, and SQL-based metadata storage for symbolic attributes. Recursive consolidation is triggered when the average pairwise semantic similarity within a memory cluster exceeds $\tau_{\text{cluster}} = 0.85$. During retrieval, we employ adaptive query-aware retrieval, where the retrieval depth is dynamically adjusted based on estimated query complexity,

Table 1. Performance on the LoCoMo benchmark with High-Capability Models (GPT-4.1 series and Qwen3-Plus). SimpleMem achieves superior efficiency-performance balance.

Model	Method	MultiHop		Temporal		OpenDomain		SingleHop		Average		Token Cost
		F1	BLEU									
GPT-4.1-mini	LoCoMo	25.02	21.62	12.04	10.63	19.05	17.07	18.68	15.87	18.70	16.30	16,910
	ReadAgent	6.48	5.6	5.31	4.23	7.66	6.62	9.18	7.91	7.16	6.09	643
	MemoryBank	5.00	4.68	5.94	4.78	5.16	4.52	5.72	4.86	5.46	4.71	432
	MemGPT	17.72	16.02	19.44	16.54	11.29	10.18	25.59	24.25	18.51	16.75	16,977
	A-Mem	25.06	17.32	51.01	44.75	13.22	14.75	41.02	36.99	32.58	28.45	2,520
	LightMem	24.96	21.66	20.55	18.39	19.21	17.68	33.79	29.66	24.63	21.85	612
	Mem0	30.14	27.62	48.91	44.82	16.43	14.94	41.3	36.17	34.20	30.89	973
GPT-4o	SimpleMem	43.46	38.82	58.62	50.10	19.76	18.04	51.12	43.53	43.24	37.62	531
	LoCoMo	28.00	18.47	9.09	5.78	16.47	14.80	61.56	54.19	28.78	23.31	16,910
	ReadAgent	14.61	9.95	4.16	3.19	8.84	8.37	12.46	10.29	10.02	7.95	805
	MemoryBank	6.49	4.69	2.47	2.43	6.43	5.30	8.28	7.10	5.92	4.88	569
	MemGPT	30.36	22.83	17.29	13.18	12.24	11.87	40.16	36.35	25.01	21.06	16,987
	A-Mem	32.86	23.76	39.41	31.23	17.10	15.84	44.43	38.97	33.45	27.45	1,216
	LightMem	28.15	21.83	36.53	29.12	13.38	11.54	33.76	28.02	27.96	22.63	645
Qwen3-Plus	Mem0	35.13	27.56	52.38	44.15	17.73	15.92	39.12	35.43	36.09	30.77	985
	SimpleMem	35.89	32.83	56.71	20.57	18.23	16.34	45.41	39.25	39.06	27.25	550
	LoCoMo	24.15	18.94	16.57	13.28	11.81	10.58	38.58	28.16	22.78	17.74	16,910
	ReadAgent	9.52	6.83	11.22	8.15	5.41	5.23	9.85	7.96	9.00	7.04	742
	MemoryBank	5.25	4.94	1.77	6.26	5.88	6.00	6.90	5.57	4.95	5.69	302
	MemGPT	25.80	17.50	24.10	18.50	9.50	7.80	40.20	42.10	24.90	21.48	16,958
	A-Mem	26.50	19.80	46.10	35.10	11.90	11.50	43.80	36.50	32.08	25.73	1,427
LongMemEval-S	LightMem	28.95	24.13	42.58	38.52	16.54	13.23	40.78	36.52	32.21	28.10	606
	Mem0	32.42	21.24	47.53	39.82	17.18	14.53	46.25	37.52	35.85	28.28	1,020
	SimpleMem	33.74	29.04	50.87	43.31	18.41	16.24	46.94	38.16	37.49	31.69	583

ranging from $k_{\min} = 3$ for simple lookups to $k_{\max} = 20$ for complex reasoning queries.

Evaluation Metrics. We report: F1 and BLEU-1 (accuracy), Adversarial Success Rate (robustness to distractors), and Token Cost (retrieval/latency efficiency). LongMemEval-S uses its standard accuracy-style metric.

3.2. Main Results and Analysis

We evaluate SimpleMem across a diverse set of LLMs, ranging from high-capability proprietary models (GPT-4o series) to efficient open-source models (Qwen series). Tables 1 and 2 present the detailed performance comparison on the LoCoMo benchmark.

Performance on High-Capability Models. As shown in Table 1, SimpleMem consistently outperforms existing memory systems across all evaluated models. On GPT-4.1-mini, SimpleMem achieves an Average F1 of 43.24, establishing a significant margin over the strongest baseline, Mem0 (34.20), and surpassing the full-context baseline (LoCoMo, 18.70) by over 24 points. Notable gains are observed in Temporal Reasoning, where SimpleMem scores 58.62 F1 compared to Mem0's 48.91, demonstrating the effectiveness of our *Semantic Structured Compression* in resolving complex timelines. Similarly, on the flagship GPT-4o, SimpleMem maintains its lead with an Average F1 of 39.06,

outperforming Mem0 (36.09) and A-Mem (33.45). These results confirm that *Recursive Consolidation* mechanism effectively distills high-density knowledge, enabling even smaller models equipped with SimpleMem to outperform larger models using traditional memory systems.

Token Efficiency. A key strength of SimpleMem lies in its inference-time efficiency. As reported in the rightmost columns of Tables 1 and 2, full-context approaches such as LoCOMO and MEMGPT consume approximately 16,900 tokens per query. In contrast, SimpleMem reduces token usage by roughly 30×, averaging 530–580 tokens per query. Furthermore, compared to optimized retrieval baselines like Mem0 (~980 tokens) and A-Mem (~1,200+ tokens), SimpleMem reduces token usage by 40–50% while delivering superior accuracy. For instance, on GPT-4.1-mini, SimpleMem uses only 531 tokens to achieve state-of-the-art performance, whereas ReadAgent consumes more (643 tokens) but achieves far lower accuracy (7.16 F1). This validates the efficacy of our *Entropy-based Filtering* and *Adaptive Pruning*, which strictly control context bandwidth without sacrificing information density.

Performance on Smaller Models. Table 2 highlights the ability of SimpleMem to empower smaller parameter models. On Qwen3-8b, SimpleMem achieves an impressive Average F1 of 33.45, significantly surpassing Mem0 (25.80) and LightMem (22.23). Crucially, a 3B-parameter model

Table 2. Performance on the LoCoMo benchmark with Efficient Models (Small parameters). SimpleMem demonstrates robust performance even on 1.5B/3B models, often surpassing larger models using baseline memory systems.

Model	Method	MultiHop F1 BLEU		Temporal F1 BLEU		OpenDomain F1 BLEU		SingleHop F1 BLEU		Average F1 BLEU		Token Cost
Qwen2.5-1.5b	LoCoMo	9.05	6.55	4.25	4.04	9.91	8.50	11.15	8.67	8.59	6.94	16,910
	ReadAgent	6.61	4.93	2.55	2.51	5.31	12.24	10.13	7.54	6.15	6.81	752
	MemoryBank	11.14	8.25	4.46	2.87	8.05	6.21	13.42	11.01	9.27	7.09	284
	MemGPT	10.44	7.61	4.21	3.89	13.42	11.64	9.56	7.34	9.41	7.62	16,953
	A-Mem	18.23	11.94	24.32	19.74	16.48	14.31	23.63	19.23	20.67	16.31	1,300
	LightMem	16.43	11.39	22.92	18.56	15.06	11.23	23.28	19.24	19.42	15.11	605
	Mem0	20.18	14.53	27.42	22.14	19.83	15.68	27.63	23.42	23.77	18.94	942
	SimpleMem	21.85	16.10	29.12	23.50	21.05	16.80	28.90	24.50	25.23	20.23	678
Qwen2.5-3b	LoCoMo	4.61	4.29	3.11	2.71	4.55	5.97	7.03	5.69	4.83	4.67	16,910
	ReadAgent	2.47	1.78	3.01	3.01	5.57	5.22	3.25	2.51	3.58	3.13	776
	MemoryBank	3.60	3.39	1.72	1.97	6.63	6.58	4.11	3.32	4.02	3.82	298
	MemGPT	5.07	4.31	2.94	2.95	7.04	7.10	7.26	5.52	5.58	4.97	16,961
	A-Mem	12.57	9.01	27.59	25.07	7.12	7.28	17.23	13.12	16.13	13.62	1,137
	LightMem	16.43	11.39	6.92	4.56	8.06	7.23	18.28	15.24	12.42	9.61	605
	Mem0	16.89	11.54	8.52	6.23	10.24	8.82	16.47	12.43	13.03	9.76	965
	SimpleMem	17.03	11.87	21.47	19.50	12.52	10.19	20.90	18.01	17.98	14.89	572
Qwen3-8b	LoCoMo	13.50	9.20	6.80	5.50	10.10	8.80	14.50	11.20	11.23	8.68	16,910
	ReadAgent	7.20	5.10	3.50	3.10	5.50	5.40	8.10	6.20	6.08	4.95	721
	MemoryBank	9.50	7.10	3.80	2.50	7.50	6.50	9.20	7.50	7.50	5.90	287
	MemGPT	14.20	9.80	5.50	4.20	12.50	10.80	11.50	9.10	10.93	8.48	16,943
	A-Mem	20.50	13.80	22.50	18.20	13.20	10.50	26.80	21.50	20.75	16.00	1,087
	LightMem	18.53	14.23	26.78	21.52	14.12	11.24	29.48	23.83	22.23	17.71	744
	Mem0	22.42	16.83	32.48	26.13	15.23	12.54	33.05	27.24	25.80	20.69	1,015
	SimpleMem	28.97	24.93	42.85	36.49	15.35	13.9	46.62	40.69	33.45	29.00	621
Qwen3-1.7b	LoCoMo	10.28	8.82	6.45	5.78	10.42	9.02	11.16	10.35	9.58	8.49	16,910
	ReadAgent	7.50	5.60	3.15	2.95	6.10	12.45	10.80	8.15	6.89	7.29	784
	MemoryBank	11.50	8.65	4.95	3.20	8.55	6.80	13.90	11.50	9.73	7.54	290
	MemGPT	11.50	8.20	4.65	4.10	13.85	11.90	10.25	7.85	10.06	8.01	16,954
	A-Mem	18.45	11.80	25.82	18.45	10.90	9.95	21.58	16.72	19.19	14.23	1,258
	LightMem	14.84	11.56	9.35	7.85	13.76	10.59	28.14	22.89	16.52	13.22	679
	Mem0	18.23	13.44	18.54	14.22	16.82	13.54	31.15	26.42	21.19	16.91	988
	SimpleMem	20.85	15.42	26.75	18.63	17.92	14.15	32.85	26.46	24.59	18.67	730

(Qwen2.5-3b) paired with SimpleMem achieves 17.98 F1, outperforming the same model with Mem0 (13.03) by nearly 5 points. Even on the extremely lightweight Qwen2.5-1.5b, SimpleMem maintains robust performance (25.23 F1), beating larger models using inferior memory strategies (e.g., Qwen3-1.7b with Mem0 scores 21.19).

Robustness Across Task Types. Breaking down performance by task, SimpleMem demonstrates balanced capabilities. In SingleHop QA, it consistently leads (e.g., 51.12 F1 on GPT-4.1-mini), proving precision in factual retrieval. In complex MultiHop scenarios, SimpleMem significantly outperforms Mem0 and LightMem on GPT-4.1-mini, indicating that our *Molecular Representations* successfully bridge disconnected facts, enabling deep reasoning without the need for expensive iterative retrieval loops.

3.3. Efficiency Analysis

We conduct a comprehensive evaluation of computational efficiency, examining both end-to-end system latency and the scalability of memory indexing and retrieval. To assess practical deployment viability, we measured the full lifecycle costs on the LoCoMo-10 dataset using GPT-4.1-mini.

As illustrated in Table 3, SimpleMem exhibits superior efficiency across all operational phases. In terms of memory construction, our system achieves the fastest processing speed at 92.6 seconds per sample. This represents a dramatic improvement over existing baselines, outperforming Mem0 by approximately 14× (1350.9s) and A-Mem by over 50× (5140.5s). This massive speedup is directly attributable to our *Semantic Structured Compression* pipeline, which processes data in a streamlined single pass, thereby avoiding the complex graph updates required by Mem0 or the iterative summarization overheads inherent to A-Mem.

Beyond construction, SimpleMem also maintains the low-

est retrieval latency at 388.3 seconds per sample, which is approximately 33% faster than LightMem and Mem0. This gain arises from the *adaptive retrieval* mechanism, which dynamically limits retrieval scope and prioritizes high-level abstract representations before accessing fine-grained details. By restricting retrieval to only the most relevant memory entries, the system avoids the expensive neighbor traversal and expansion operations that commonly dominate the latency of graph-based memory systems.

When considering the total time-to-insight, SimpleMem achieves a 4 \times speedup over Mem0 and a 12 \times speedup over A-Mem. Crucially, this efficiency does not come at the expense of performance. On the contrary, SimpleMem achieves the highest Average F1 among all compared methods. These results support our central claim that structured semantic compression and adaptive retrieval produce a more compact and effective reasoning substrate than raw context retention or graph-centric memory designs, enabling a superior balance between accuracy and computational efficiency.

Table 3. Comparison of construction time, retrieval time, total experiment time, and average F1 score across different memory systems (tested on LoCoMo-10 with GPT-4.1-mini).

Model	Construction Time	Retrieve Time	Total Time	Average F1
A-mem	5140.5s	796.7s	5937.2s	32.58
Lightmem	97.8s	577.1s	675.9s	24.63
Mem0	1350.9s	583.4s	1934.3s	34.20
SimpleMem	92.6s	388.3s	480.9s	43.24

3.4. Ablation Study

To verify the claims that specific cognitive mechanisms correspond to computational gains, we conducted a component-wise ablation study using the GPT-4.1-mini backend. We investigate the contribution of three key components: (1) *Semantic Structured Compression*, (2) *Recursive Consolidation*, and (3) *Adaptive Query-Aware Retrieval*. The results are summarized in Table 4.

Impact of Semantic Structured Compression. Replacing the proposed compression pipeline with standard chunk-based storage leads to a substantial degradation in temporal reasoning performance. Specifically, removing semantic structured compression reduces the Temporal F1 by 56.7%, from 58.62 to 25.40. This drop indicates that without context normalization steps such as resolving coreferences and converting relative temporal expressions into absolute timestamps, the retriever struggles to disambiguate events along the timeline. As a result, performance regresses to levels comparable to conventional retrieval-augmented generation systems that rely on raw or weakly structured context.

Impact of Recursive Consolidation. Disabling the background consolidation process results in a 31.3% decrease in multi-hop reasoning performance. Without consolidat-

ing related memory units into higher-level abstract representations, the system must retrieve a larger number of fragmented entries during reasoning. This fragmentation increases context redundancy and exhausts the available context window in complex queries, demonstrating that recursive consolidation is essential for synthesizing dispersed evidence into compact and informative representations.

Impact of Adaptive Query-Aware Retrieval. Removing the adaptive retrieval mechanism and reverting to fixed-depth retrieval primarily degrades performance on open-domain and single-hop tasks, with drops of 26.6% and 19.4%, respectively. In the absence of query-aware adjustment, the system either retrieves insufficient context for entity-specific queries or introduces excessive irrelevant information for simple queries. These results highlight the importance of dynamically modulating retrieval scope to balance relevance and efficiency during inference.

3.5. Case Study: Long-Term Temporal Grounding

To illustrate how SimpleMem handles long-horizon conversational history, Figure 3 presents a representative multi-session example spanning two weeks and approximately 24,000 raw tokens. SimpleMem filters low-information dialogue during ingestion and retains only high-utility memory entries, reducing the stored memory to about 800 tokens without losing task-relevant content.

Temporal Normalization. Relative temporal expressions such as “last week” and “yesterday” refer to different absolute times across sessions. SimpleMem resolves it into absolute timestamps at memory construction time, ensuring consistent temporal grounding over long interaction gaps.

Precise Retrieval. When queried about Sarah’s past artworks, the adaptive retrieval mechanism combines semantic relevance with symbolic constraints to exclude unrelated activities and retrieve only temporally valid entries. The system correctly identifies relevant paintings while ignoring semantically related but irrelevant topics. This example demonstrates how structured compression, temporal normalization, and adaptive retrieval jointly enable reliable long-term reasoning under extended interaction histories.

4. Related Work

Memory Systems for LLM Agents. Recent approaches manage memory through virtual context or structured representations. Virtual context methods, including MEMGPT (Packer et al., 2023), MEMORYOS (Kang et al., 2025), and SCM (Wang et al., 2023), extend interaction length via paging or stream-based controllers (Wang et al., 2024b) but typically store raw conversation logs, leading to redundancy and increasing processing costs. In parallel, structured and graph-based systems, such as MEMORYBANK (Zhong et al.,

Table 4. Full Ablation Analysis with GPT-4.1-mini backend. The "Diff" columns indicate the percentage drop relative to the full SimpleMem model. The results confirm that each stage contributes significantly to specific reasoning capabilities.

Configuration	Multi-hop F1		Temporal F1		Open Domain F1		Single Hop F1		Average F1	
	F1	Diff	F1	Diff	F1	Diff	F1	Diff	F1	Diff
Full SimpleMem	43.46	-	58.62	-	19.76	-	51.12	-	43.24	-
w/o Atomization	34.20	(↓21.3%)	25.40	(↓56.7%)	17.50	(↓11.4%)	48.05	(↓6.0%)	31.29	(↓27.6%)
w/o Consolidation	29.85	(↓31.3%)	55.10	(↓6.0%)	18.20	(↓7.9%)	49.80	(↓2.6%)	38.24	(↓11.6%)
w/o Adaptive Pruning	38.60	(↓11.2%)	56.80	(↓3.1%)	14.50	(↓26.6%)	41.20	(↓19.4%)	37.78	(↓12.6%)

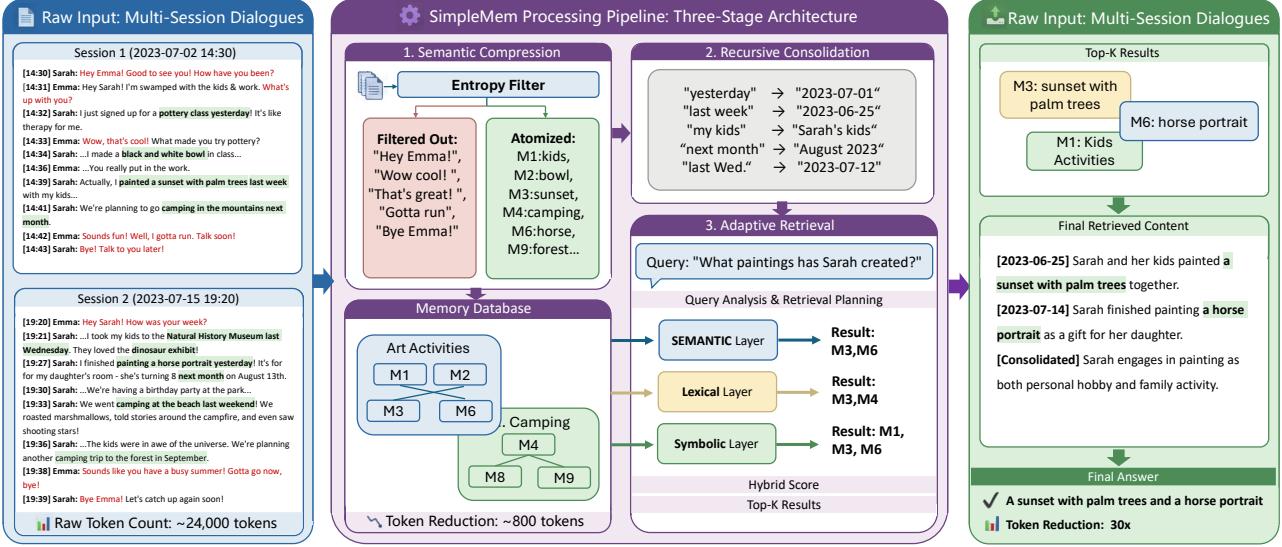


Figure 3. A Case of SimpleMem for Long-Term Multi-Session Dialogues. SimpleMem processes multi-session dialogues by filtering redundant content, normalizing temporal references, and organizing memories into compact representations. During retrieval, it adaptively combines semantic, lexical, and symbolic signals to select relevant entries.

2024), MEMO (Dev & Taraneet, 2024), ZEP (Rasmussen et al., 2025), A-MEM (Xu et al., 2025), and O-MEM (Wang et al., 2025), impose structural priors to improve coherence but still rely on raw or minimally processed text, preserving referential and temporal ambiguities that degrade long-term retrieval. In contrast, SimpleMem adopts a semantic compression mechanism that converts dialogue into independent, self-contained facts, explicitly resolving referential and temporal ambiguities prior to storage.

Context Management and Retrieval Efficiency. Beyond memory storage, efficient access to historical information remains a core challenge. Existing approaches primarily rely on either long-context models or retrieval-augmented generation (RAG). Although recent LLMs support extended context windows (OpenAI, 2025; Deepmind, 2025; Anthropic, 2025), and prompt compression methods aim to reduce costs (Jiang et al., 2023a; Liskovetsky et al., 2025), empirical studies reveal the “Lost-in-the-Middle” effect (Liu et al., 2023; Kuratov et al., 2024), where reasoning performance degrades as context length increases, alongside prohibitive computational overhead for lifelong agents. RAG-based methods (Lewis et al., 2020; Asai et al., 2023; Jiang et al., 2023b), including structurally enhanced vari-

ants such as GRAPHRAG (Edge et al., 2024; Zhao et al., 2025) and LIGHTRAG (Guo et al., 2024), decouple memory from inference but are largely optimized for static knowledge bases, limiting their effectiveness for dynamic, time-sensitive episodic memory. In contrast, SimpleMem improves retrieval efficiency through Adaptive Pruning and Retrieval, jointly leveraging semantic, lexical, and metadata signals to enable precise filtering by entities and timestamps, while dynamically adjusting retrieval depth based on query complexity to minimize token usage.

5. Conclusion

We introduce SimpleMem, an efficient memory architecture governed by the principle of Semantic Lossless Compression. By reimagining memory as a metabolic process, SimpleMem implements a dynamic continuum: *Semantic Structured Compression* to filter noise at the source, *Recursive Consolidation* to evolve fragmented facts into high-order molecular insights, and *Adaptive Spatial Pruning* to dynamically modulate retrieval bandwidth. Empirical evaluation on the LoCoMo benchmark demonstrates the effectiveness and efficiency of SimpleMem.

Acknowledgement

This work is partially supported by Amazon Research Award, Cisco Faculty Research Award, and Coefficient Giving.

References

- Anthropic. Claude 3.7 sonnet and claude code. <https://www.anthropic.com/news/clause-3-7-sonnet>, 2025.
- Asai, A., Wu, Z., Wang, Y., Sil, A., and Hajishirzi, H. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*, 2023.
- Deepmind, G. Gemini 2.5: Our most intelligent AI model — blog.google. <https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/#gemini-2-5-thinking>, 2025. Accessed: 2025-03-25.
- Dev, K. and Taranjeet, S. mem0: The memory layer for ai agents. <https://github.com/mem0ai/mem0>, 2024.
- Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., Truitt, S., and Larson, J. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.
- Fang, J., Deng, X., Xu, H., Jiang, Z., Tang, Y., Xu, Z., Deng, S., Yao, Y., Wang, M., Qiao, S., et al. Lightmem: Lightweight and efficient memory-augmented generation. *arXiv preprint arXiv:2510.18866*, 2025.
- Guo, Z., Xia, L., Yu, Y., Ao, T., and Huang, C. Lightrag: Simple and fast retrieval-augmented generation. *arXiv preprint arXiv:2410.05779*, 2024.
- Hu, Y., Liu, S., Yue, Y., Zhang, G., Liu, B., Zhu, F., Lin, J., Guo, H., Dou, S., Xi, Z., et al. Memory in the age of ai agents. *arXiv preprint arXiv:2512.13564*, 2025.
- Jiang, H., Wu, Q., Lin, C.-Y., Yang, Y., and Qiu, L. Llmlingua: Compressing prompts for accelerated inference of large language models. *arXiv preprint arXiv:2310.05736*, 2023a.
- Jiang, Z., Xu, F. F., Gao, L., Sun, Z., Liu, Q., Dwivedi-Yu, J., Yang, Y., Callan, J., and Neubig, G. Active retrieval augmented generation. *arXiv preprint arXiv:2305.06983*, 2023b.
- Kang, J., Ji, M., Zhao, Z., and Bai, T. Memory os of ai agent. *arXiv preprint arXiv:2506.06326*, 2025.
- Kumaran, D., Hassabis, D., and McClelland, J. L. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in cognitive sciences*, 20(7):512–534, 2016.
- Kuratov, Y. et al. In case of context: Investigating the effects of long context on language model performance. *arXiv preprint*, 2024.
- Lee, K.-H., Chen, X., Furuta, H., Canny, J., and Fischer, I. A human-inspired reading agent with gist memory of very long contexts. *arXiv preprint arXiv:2402.09727*, 2024.
- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktaschel, T., et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.
- Li, Z., Song, S., Wang, H., Niu, S., Chen, D., Yang, J., Xi, C., Lai, H., Zhao, J., Wang, Y., Ren, J., Lin, Z., Huo, J., Chen, T., Chen, K., Li, K.-R., Yin, Z., Yu, Q., Tang, B., Yang, H., Xu, Z., and Xiong, F. Memos: An operating system for memory-augmented generation (mag) in large language models. *ArXiv*, abs/2505.22101, 2025. URL <https://api.semanticscholar.org/CorpusID:278960153>.
- Liskavetsky, A. et al. Compressor: Context-aware prompt compression for enhanced llm inference. *arXiv preprint*, 2025.
- Liu, J., Xiong, K., Xia, P., Zhou, Y., Ji, H., Feng, L., Han, S., Ding, M., and Yao, H. Agent0-vl: Exploring self-evolving agent for tool-integrated vision-language reasoning. *arXiv preprint arXiv:2511.19900*, 2025.
- Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023.
- Maharana, A., Lee, D.-H., Tulyakov, S., Bansal, M., Barbieri, F., and Fang, Y. Evaluating very long-term conversational memory of llm agents, 2024. URL <https://arxiv.org/abs/2402.17753>.
- OpenAI. Introducing gpt-5. <https://openai.com/index/introducing-gpt-5/>, 2025.
- Ouyang, S., Yan, J., Hsu, I., Chen, Y., Jiang, K., Wang, Z., Han, R., Le, L. T., Daruki, S., Tang, X., et al. Reasoningbank: Scaling agent self-evolving with reasoning memory. *arXiv preprint arXiv:2509.25140*, 2025.
- Packer, C., Fang, V., Patil, S. G., Lin, K., Wooders, S., and Gonzalez, J. Memgpt: Towards llms as operating systems. *ArXiv*, abs/2310.08560,

2023. URL <https://api.semanticscholar.org/CorpusID:263909014>.
- Qiu, J., Qi, X., Zhang, T., Juan, X., Guo, J., Lu, Y., Wang, Y., Yao, Z., Ren, Q., Jiang, X., et al. Alita: Generalist agent enabling scalable agentic reasoning with minimal predefinition and maximal self-evolution. *arXiv preprint arXiv:2505.20286*, 2025.
- Rasmussen, P., Paliychuk, P., Beauvais, T., Ryan, J., and Chalef, D. Zep: a temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956*, 2025.
- Tang, X., Qin, T., Peng, T., Zhou, Z., Shao, D., Du, T., Wei, X., Xia, P., Wu, F., Zhu, H., et al. Agent kb: Leveraging cross-domain experience for agentic problem solving. *arXiv preprint arXiv:2507.06229*, 2025.
- Team, T. D., Li, B., Zhang, B., Zhang, D., Huang, F., Li, G., Chen, G., Yin, H., Wu, J., Zhou, J., et al. Tongyi deepresearch technical report. *arXiv preprint arXiv:2510.24701*, 2025.
- Tu, A., Xuan, W., Qi, H., Huang, X., Zeng, Q., Talaei, S., Xiao, Y., Xia, P., Tang, X., Zhuang, Y., et al. Position: The hidden costs and measurement gaps of reinforcement learning with verifiable rewards. *arXiv preprint arXiv:2509.21882*, 2025.
- Wang, B., Liang, X., Yang, J., Huang, H., Wu, S., Wu, P., Lu, L., Ma, Z., and Li, Z. Enhancing large language model with self-controlled memory framework. *arXiv preprint arXiv:2304.13343*, 2023.
- Wang, P., Tian, M., Li, J., Liang, Y., Wang, Y., Chen, Q., Wang, T., Lu, Z., Ma, J., Jiang, Y. E., et al. O-mem: Omni memory system for personalized, long horizon, self-evolving agents. *arXiv e-prints*, pp. arXiv–2511, 2025.
- Wang, T., Tao, M., Fang, R., Wang, H., Wang, S., Jiang, Y. E., and Zhou, W. Ai persona: Towards life-long personalization of llms. *arXiv preprint arXiv:2412.13103*, 2024a.
- Wang, Y. and Chen, X. Mirix: Multi-agent memory system for llm-based agents. *arXiv preprint arXiv:2507.07957*, 2025.
- Wang, Z. Z., Mao, J., Fried, D., and Neubig, G. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024b.
- Xia, P., Zeng, K., Liu, J., Qin, C., Wu, F., Zhou, Y., Xiong, C., and Yao, H. Agent0: Unleashing self-evolving agents from zero data via tool-integrated reasoning. *arXiv preprint arXiv:2511.16043*, 2025.
- Xu, W., Liang, Z., Mei, K., Gao, H., Tan, J., and Zhang, Y. A-mem: Agentic memory for llm agents. *ArXiv*, abs/2502.12110, 2025. URL <https://api.semanticscholar.org/CorpusID:276421617>.
- Yan, B., Li, C., Qian, H., Lu, S., and Liu, Z. General agentic memory via deep research. *arXiv preprint arXiv:2511.18423*, 2025.
- Yang, B., Xu, L., Zeng, L., Liu, K., Jiang, S., Lu, W., Chen, H., Jiang, X., Xing, G., and Yan, Z. Contextagent: Context-aware proactive llm agents with open-world sensory perceptions. *arXiv preprint arXiv:2505.14668*, 2025.
- Zhao, Y., Zhu, J., Guo, Y., He, K., and Li, X. E^2graphrag: Streamlining graph-based rag for high efficiency and effectiveness. *arXiv preprint arXiv:2505.24226*, 2025.
- Zhong, W., Guo, L., Gao, Q., Ye, H., and Wang, Y. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19724–19731, 2024.

A. Detailed System Prompts

To ensure full reproducibility of the **SimpleMem** pipeline, we provide the exact system prompts used in the key processing stages. All prompts are designed to be model-agnostic but were optimized for GPT-4o-mini in our experiments to ensure cognitive economy.

A.1. Stage 1: Semantic Structured Compression Prompt

This prompt performs entropy-aware filtering and context normalization. Its goal is to transform raw dialogue windows into compact, context-independent memory units while excluding low-information interaction content.

Listing 1. Prompt for Semantic Structured Compression and Normalization.

```
You are a memory encoder in a long-term
memory system. Your task is to
transform raw conversational input into
compact, self-contained memory units.

INPUT METADATA:
Window Start Time: {window_start_time} (ISO
8601)
Participants: {speakers_list}

INSTRUCTIONS:
1. Information Filtering:
   - Discard social filler,
     acknowledgements, and conversational
     routines that introduce no new
     factual or semantic information.
   - Discard redundant confirmations unless
     they modify or finalize a decision.
   - If no informative content is present,
     output an empty list.

2. Context Normalization:
   - Resolve all pronouns and implicit
     references into explicit entity
     names.
   - Ensure each memory unit is
     interpretable without access to
     prior dialogue.

3. Temporal Normalization:
   - Convert relative temporal expressions
     (e.g., "tomorrow", "last week") into
     absolute ISO 8601 timestamps using
     the window start time.

4. Memory Unit Extraction:
   - Decompose complex utterances into
     minimal, indivisible factual
     statements.

INPUT DIALOGUE:
{dialogue_window}

OUTPUT FORMAT (JSON):
{
  "memory_units": [
    {
      "content": "Alice agreed to meet Bob
      at the Starbucks on 5th Avenue on
      2025-11-20T14:00:00.",
      "entities": ["Alice", "Bob", "Starbucks", "5th Avenue"],
      "topic": "Meeting Planning",
      "timestamp": "2025-11-20T14:00:00",
      "salience": "high"
    }
  ]
}
```

```
{
  "content": "Alice agreed to meet Bob
  at the Starbucks on 5th Avenue on
  2025-11-20T14:00:00.",
  "entities": ["Alice", "Bob", "Starbucks", "5th Avenue"],
  "topic": "Meeting Planning",
  "timestamp": "2025-11-20T14:00:00",
  "salience": "high"
}
]
```

A.2. Stage 2: Adaptive Retrieval Planning Prompt

This prompt analyzes the user query prior to retrieval. Its purpose is to estimate query complexity and generate a structured retrieval plan that adapts retrieval scope accordingly.

Listing 2. Prompt for Query Analysis and Adaptive Retrieval Planning.

```
Analyze the following user query and
generate a retrieval plan. Your
objective is to retrieve sufficient
information while minimizing
unnecessary context usage.

USER QUERY:
{user_query}

INSTRUCTIONS:
1. Query Complexity Estimation:
   - Assign "LOW" if the query can be
     answered via direct fact lookup or a
     single memory unit.
   - Assign "HIGH" if the query requires
     aggregation across multiple events,
     temporal comparison, or synthesis of
     patterns.

2. Retrieval Signals:
   - Lexical layer: extract exact keywords
     or entity names.
   - Temporal layer: infer absolute time
     ranges if relevant.
   - Semantic layer: rewrite the query into
     a declarative form suitable for
     semantic matching.

OUTPUT FORMAT (JSON):
{
  "complexity": "HIGH",
  "retrieval_rationale": "The query
    requires reasoning over multiple
    temporally separated events.",
  "lexical_keywords": ["Starbucks", "Bob"],
  "temporal_constraints": {
    "start": "2025-11-01T00:00:00",
    "end": "2025-11-30T23:59:59"
  },
  "semantic_query": "The user is asking
    about the scheduled meeting with Bob,
    including location and time."
}
```

```
}
```

A.3. Stage 3: Reconstructive Synthesis Prompt

This prompt guides the final answer generation using retrieved memory. It combines high-level abstract representations with fine-grained factual details to produce a grounded response.

Listing 3. Prompt for Reconstructive Synthesis (Answer Generation).

```
You are an assistant with access to a
structured long-term memory.

USER QUERY:
{user_query}

RETRIEVED MEMORY (Ordered by Relevance):

[ABSTRACT REPRESENTATIONS]:
{retrieved_abstracts}

[DETAILED MEMORY UNITS]:
{retrieved_units}

INSTRUCTIONS:
1. Hierarchical Reasoning:
   - Use abstract representations to
     capture recurring patterns or
     general user preferences.
   - Use detailed memory units to ground
     the response with specific facts.

2. Conflict Handling:
   - If inconsistencies arise, prioritize
     the most recent memory unit.
   - Optionally reference abstract patterns
     when relevant.

3. Temporal Consistency:
   - Ensure all statements respect the
     timestamps provided in memory.

4. Faithfulness:
   - Base the answer strictly on the
     retrieved memory.
   - If required information is missing,
     respond with: "I do not have enough
     information in my memory."

FINAL ANSWER:
```

particular attention to the thresholds governing semantic structured compression and recursive consolidation.

B.2. Hyperparameter Sensitivity Analysis

To assess the effectiveness of semantic structured compression and to motivate the design of adaptive retrieval, we analyze system sensitivity to the number of retrieved memory entries (k). We vary k from 1 to 20 and report the average F1 score on the LoCoMo benchmark using the GPT-4.1-mini backend.

Table 5. Performance sensitivity to retrieval count (k). **SimpleMem** demonstrates "Rapid Saturation," reaching near-optimal performance at $k = 3$ (42.85) compared to its peak at $k = 10$ (43.45). This validates the high information density of Atomic Entries, proving that huge context windows are often unnecessary for accuracy.

Method	Top- k Retrieved Entries				
	$k=1$	$k=3$	$k=5$	$k=10$	$k=20$
ReadAgent	6.12	8.45	9.18	8.92	8.50
MemGPT	18.40	22.15	25.59	24.80	23.10
SimpleMem	35.20	42.85	43.24	43.45	43.40

Table 5 provides two key observations. First, rapid performance saturation is observed at low retrieval depth. SimpleMem achieves strong performance with a single retrieved entry (35.20 F1) and reaches approximately 99% of its peak performance at $k = 3$. This behavior indicates that semantic structured compression produces memory units with high information content, often sufficient to answer a query without aggregating many fragments.

Second, robustness to increased retrieval depth distinguishes SimpleMem from baseline methods. While approaches such as MemGPT experience performance degradation at larger k , SimpleMem maintains stable accuracy even when retrieving up to 20 entries. This robustness enables adaptive retrieval to safely expand context for complex reasoning tasks without introducing excessive irrelevant information.

B. Extended Implementation Details and Experiments

B.1. Hyperparameter Configuration

Table 6 summarizes the hyperparameters used to obtain the results reported in Section 3. These values were selected to balance memory compactness and retrieval recall, with

Table 6. Detailed hyperparameter configuration for SimpleMem. The system employs adaptive thresholds to balance memory compactness and retrieval effectiveness.

Module	Parameter	Value / Description
<i>Stage 1: Semantic Structured Compression</i>	Window Size (W)	10 turns
	Sliding Stride	5 turns (50% overlap)
	Information Threshold (τ)	0.35 (filters low-information interaction content)
	Model Backend	gpt-4o-mini (temperature = 0.0)
	Coreference Scope	Current window with up to two preceding turns
	Output Constraint	Strict JSON schema enforced
<i>Stage 2: Recursive Consolidation</i>	Embedding Model	text-embedding-3-small (1536 dimensions)
	Consolidation Threshold (τ_{cluster})	0.85 (triggers abstraction over related memory units)
	Temporal Decay (λ)	0.1 (controls temporal influence during consolidation)
	Vector Database	LanceDB (v0.4.5) with IVF-PQ indexing
<i>Stage 3: Adaptive Retrieval</i>	Stored Metadata	timestamp, entities, topic, salience
	Query Complexity Estimator	gpt-4o-mini (classification head)
	Retrieval Range	$k \in [3, 20]$
	Minimum Depth (k_{\min})	3 (symbolic and abstract-level retrieval)
	Maximum Depth (k_{\max})	20 (expanded semantic retrieval)
	Re-ranking	Disabled (multi-view score fusion applied directly)