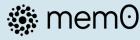


# Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav

[research@mem0.ai](mailto:research@mem0.ai)



Large Language Models (LLMs) have demonstrated remarkable prowess in generating contextually coherent responses, yet their fixed context windows pose fundamental challenges for maintaining consistency over prolonged multi-session dialogues. We introduce Mem0, a scalable memory-centric architecture that addresses this issue by dynamically extracting, consolidating, and retrieving salient information from ongoing conversations. Building on this foundation, we further propose an enhanced variant that leverages graph-based memory representations to capture complex relational structures among conversational elements. Through comprehensive evaluations on the LOCOMO benchmark, we systematically compare our approaches against six baseline categories: (i) established memory-augmented systems, (ii) retrieval-augmented generation (RAG) with varying chunk sizes and  $k$ -values, (iii) a full-context approach that processes the entire conversation history, (iv) an open-source memory solution, (v) a proprietary model system, and (vi) a dedicated memory management platform. Empirical results demonstrate that our methods consistently outperform all existing memory systems across four question categories: single-hop, temporal, multi-hop, and open-domain. Notably, Mem0 achieves 26% relative improvements in the LLM-as-a-Judge metric over OpenAI, while Mem0 with graph memory achieves around 2% higher overall score than the base Mem0 configuration. Beyond accuracy gains, we also markedly reduce computational overhead compared to the full-context approach. In particular, Mem0 attains a 91% lower p95 latency and saves more than 90% token cost, thereby offering a compelling balance between advanced reasoning capabilities and practical deployment constraints. Our findings highlight the critical role of structured, persistent memory mechanisms for long-term conversational coherence, paving the way for more reliable and efficient LLM-driven AI agents.

Code can be found at: <https://mem0.ai/research>

## 1. Introduction

Human memory is a *foundation of intelligence*—it shapes our identity, guides decision-making, and enables us to learn, adapt, and form meaningful relationships (Craik and Jennings, 1992). Among its many roles, memory is essential for communication: we recall past interactions, infer preferences, and construct evolving mental models of those we engage with (Assmann, 2011). This ability to retain and retrieve information over extended periods enables coherent, contextually rich exchanges that span days, weeks, or even months. AI agents, powered by large language models (LLMs), have made remarkable progress in generating fluent, contextually appropriate responses (Yu et al., 2024, Zhang et al., 2024). However, these systems are fundamentally limited by their reliance on fixed context windows, which severely restrict their ability to maintain coherence over extended interactions (Bulatov et al., 2022, Liu et al., 2023). This limitation stems from LLMs' lack of persistent memory mechanisms that can extend beyond their finite context windows. While humans naturally accumulate and organize experiences over time, forming a continuous narrative



degrade over distant tokens (Guo et al., 2024, Nelson et al., 2024). This limitation is particularly problematic in high-stakes domains such as healthcare, education, and enterprise support, where maintaining continuity and trust is crucial (Hatalis et al., 2023). To address these challenges, AI agents must adopt memory systems that go beyond static context extension. A robust AI memory should selectively store important information, consolidate related concepts, and retrieve relevant details when needed—mirroring human cognitive processes (He et al., 2024). By integrating such mechanisms, we can develop AI agents that maintain consistent personas, track evolving user preferences, and build upon prior exchanges. This shift will transform AI from transient, forgetful responders into reliable, long-term collaborators, fundamentally redefining the future of conversational intelligence.

In this paper, we address a fundamental limitation in AI systems: their inability to maintain coherent reasoning across extended conversations across different sessions, which severely restricts meaningful long-term interactions with users. We introduce Mem0 (pronounced as *mem-zero*), a novel memory architecture that dynamically captures, organizes, and retrieves salient information from ongoing conversations. Building on this foundation, we develop Mem0<sup>g</sup>, which enhances the base architecture with graph-based memory representations to better model complex relationships between conversational elements. Our experimental results on the LOCOMO benchmark demonstrate that our approaches consistently outperform existing memory systems—including memory-augmented architectures, retrieval-augmented generation (RAG) methods, and both open-source and proprietary solutions—across diverse question types, while simultaneously requiring significantly lower computational resources. Latency measurements further reveal that Mem0 operates with 91% lower response times than full-context approaches, striking an optimal balance between sophisticated reasoning capabilities and practical deployment constraints. These contributions represent a meaningful step toward AI systems that can maintain coherent, context-aware conversations over extended durations—mirroring human communication patterns and opening new possibilities for applications in personal tutoring, healthcare, and personalized assistance.

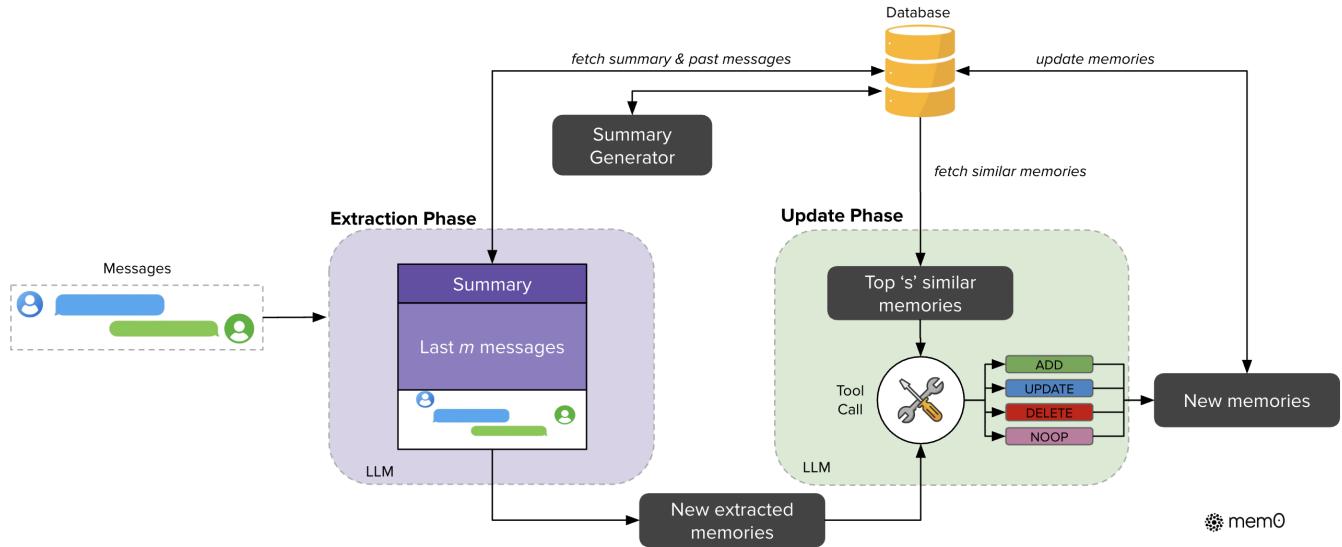
## 2. Proposed Methods

We introduce two memory architectures for AI agents. (1) Mem0 implements a novel paradigm that extracts, evaluates, and manages salient information from conversations through dedicated modules for memory extraction and updation. The system processes a pair of messages between either two user participants or a user and an assistant. (2) Mem0<sup>g</sup> extends this foundation by incorporating graph-based memory representations, where memories are stored as directed labeled graphs with entities as nodes and relationships as edges. This structure enables a deeper understanding of the connections between entities. By explicitly modeling both entities and their relationships, Mem0<sup>g</sup> supports more advanced reasoning across interconnected facts, especially for queries that require navigating complex relational paths across multiple memories.

### 2.1. Mem0

Our architecture follows an incremental processing paradigm, enabling it to operate seamlessly within ongoing conversations. As illustrated in Figure 2, the complete pipeline architecture consists of two phases: *extraction* and *update*.

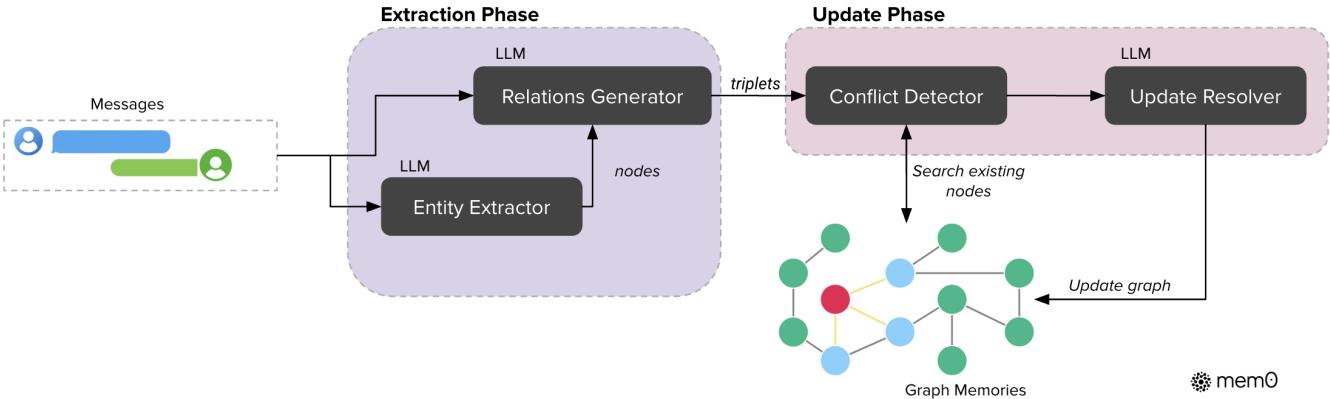
The **extraction phase** initiates upon **ingestion of a new message pair** ( $m_{t-1}, m_t$ ), where  $m_t$  represents the current message and  $m_{t-1}$  the preceding one. This pair typically consists of a user message and an assistant response, capturing a complete interaction unit. To establish appropriate context for memory extraction, the system employs two complementary sources: (1) **a conversation summary  $S$  retrieved from the database** that



**Figure 2:** Architectural overview of the MemO system showing extraction and update phase. The extraction phase processes messages and historical context to create new memories. The update phase evaluates these extracted memories against similar existing ones, applying appropriate operations through a Tool Call mechanism. The database serves as the central repository, providing context for processing and storing updated memories.

encapsulates the semantic content of the entire conversation history, and (2) a sequence of recent messages  $\{m_{t-m}, m_{t-m+1}, \dots, m_{t-2}\}$  from the conversation history, where  $m$  is a hyperparameter controlling the recency window. To support context-aware memory extraction, we implement an asynchronous summary generation module that periodically refreshes the conversation summary. This component operates independently of the main processing pipeline, ensuring that memory extraction consistently benefits from up-to-date contextual information without introducing processing delays. While  $S$  provides global thematic understanding across the entire conversation, the recent message sequence offers granular temporal context that may contain relevant details not consolidated in the summary. This dual contextual information, combined with the new message pair, forms a comprehensive prompt  $P = (S, \{m_{t-m}, \dots, m_{t-2}\}, m_{t-1}, m_t)$  for an extraction function  $\phi$  implemented via an LLM. The function  $\phi(P)$  then extracts a set of salient memories  $\Omega = \{\omega_1, \omega_2, \dots, \omega_n\}$  specifically from the new exchange while maintaining awareness of the conversation's broader context, resulting in candidate facts for potential inclusion in the knowledge base.

Following extraction, the **update phase** evaluates each candidate fact against existing memories to maintain consistency and avoid redundancy. This phase determines the appropriate memory management operation for each extracted fact  $\omega_i \in \Omega$ . Algorithm 1, mentioned in Appendix B, illustrates this process. For each fact, the system first retrieves the top  $s$  semantically similar memories using vector embeddings from the database. These retrieved memories, along with the candidate fact, are then presented to the LLM through a function-calling interface we refer to as a ‘tool call.’ The LLM itself determines which of four distinct operations to execute: **ADD** for creation of new memories when no semantically equivalent memory exists; **UPDATE** for augmentation of existing memories with complementary information; **DELETE** for removal of memories contradicted by new information; and **NOOP** when the candidate fact requires no modification to the knowledge base. Rather than using a separate classifier, we leverage the LLM’s reasoning capabilities to directly select the appropriate operation based on the semantic relationship between the candidate fact and existing memories. Following this determination, the system executes the provided operations, thereby maintaining knowledge base coherence and temporal consistency.



**Figure 3:** Graph-based memory architecture of Mem0<sup>g</sup> illustrating entity extraction and update phase. The extraction phase uses LLMs to convert conversation messages into entities and relation triplets. The update phase employs conflict detection and resolution mechanisms when integrating new information into the existing knowledge graph.

In our experimental evaluation, we configured the system with ‘ $m$ ’ = 10 previous messages for contextual reference and ‘ $s$ ’ = 10 similar memories for comparative analysis. All language model operations utilized GPT-4o-mini as the inference engine. The vector database employs dense embeddings to facilitate efficient similarity search during the update phase.

## 2.2. Mem0<sup>g</sup>

The Mem0<sup>g</sup> pipeline, illustrated in Figure 3, implements a graph-based memory approach that effectively captures, stores, and retrieves contextual information from natural language interactions (Zhang et al., 2022). In this framework, memories are represented as a directed labeled graph  $G = (V, E, L)$ , where:

- Nodes  $V$  represent entities (e.g., ALICE, SAN\_FRANCISCO)
- Edges  $E$  represent relationships between entities (e.g., LIVES\_IN)
- Labels  $L$  assign semantic types to nodes (e.g., ALICE - Person, SAN\_FRANCISCO - City)

Each entity node  $v \in V$  contains three components: (1) an entity type classification that categorizes the entity (e.g., Person, Location, Event), (2) an embedding vector  $e_v$  that captures the entity’s semantic meaning, and (3) metadata including a creation timestamp  $t_v$ . Relationships in our system are structured as triplets in the form  $(v_s, r, v_d)$ , where  $v_s$  and  $v_d$  are source and destination entity nodes, respectively, and  $r$  is the labeled edge connecting them.

The extraction process employs a two-stage pipeline leveraging LLMs to transform unstructured text into structured graph representations. First, an *entity extractor* module processes the input text to identify a set of entities along with their corresponding types. In our framework, entities represent the key information elements in conversations—including people, locations, objects, concepts, events, and attributes that merit representation in the memory graph. The entity extractor identifies these diverse information units by analyzing the semantic importance, uniqueness, and persistence of elements in the conversation. For instance, in a conversation about travel plans, entities might include destinations (cities, countries), transportation modes, dates, activities, and participant preferences—essentially any discrete information that could be relevant for future reference or reasoning.



Next, a **relationship generator** component derives meaningful connections between these entities, establishing a set of relationship triplets that capture the semantic structure of the information. This LLM-based module analyzes the extracted entities and their context within the conversation to identify semantically significant connections. It works by examining linguistic patterns, contextual cues, and domain knowledge to determine how entities relate to one another. For each potential entity pair, the generator evaluates whether a meaningful relationship exists and, if so, classifies this relationship with an appropriate label (e.g., ‘lives\_in’, ‘prefers’, ‘owns’, ‘happened\_on’). The module employs prompt engineering techniques that guide the LLM to reason about both explicit statements and implicit information in the dialogue, resulting in relationship triplets that form the edges in our memory graph and enable complex reasoning across interconnected information. When integrating new information, Mem0<sup>g</sup> employs a sophisticated storage and update strategy. For each new relationship triple, we compute embeddings for both source and destination entities, then search for existing nodes with semantic similarity above a defined threshold ‘ $r$ ’. Based on node existence, the system may create both nodes, create only one node, or use existing nodes before establishing the relationship with appropriate metadata. To maintain a consistent knowledge graph, we implement a **conflict detection** mechanism that identifies potentially conflicting existing relationships when new information arrives. An LLM-based **update resolver** determines if certain relationships should be obsolete, marking them as invalid rather than physically removing them to enable temporal reasoning.

The memory retrieval functionality in Mem0<sup>g</sup> implements a dual-approach strategy for optimal information access. The entity-centric method first identifies key entities within a query, then leverages semantic similarity to locate corresponding nodes in the knowledge graph. It systematically explores both incoming and outgoing relationships from these anchor nodes, constructing a comprehensive subgraph that captures relevant contextual information. Complementing this, the semantic triplet approach takes a more holistic view by encoding the entire query as a dense embedding vector. This query representation is then matched against textual encodings of each relationship triplet in the knowledge graph. The system calculates fine-grained similarity scores between the query and all available triplets, returning only those that exceed a configurable relevance threshold, ranked in order of decreasing similarity. This dual retrieval mechanism enables Mem0<sup>g</sup> to handle both targeted entity-focused questions and broader conceptual queries with equal effectiveness.

From an implementation perspective, the system utilizes Neo4j<sup>1</sup> as the underlying graph database. LLM-based extractors and update module leverage GPT-4o-mini with function calling capabilities, allowing for structured extraction of information from unstructured text. By combining graph-based representations with semantic embeddings and LLM-based information extraction, Mem0<sup>g</sup> achieves both the structural richness needed for complex reasoning and the semantic flexibility required for natural language understanding.

### 3. Experimental Setup

#### 3.1. Dataset

The LOCOMO (Maharana et al., 2024) dataset is designed to evaluate long-term conversational memory in dialogue systems. It comprises 10 extended conversations, each containing approximately 600 dialogues and 26000 tokens on average, distributed across multiple sessions. Each conversation captures two individuals discussing daily experiences or past events. Following these multi-session dialogues, each conversation is accompanied by 200 questions on an average with corresponding ground truth answers. These questions are categorized into multiple types: single-hop, multi-hop, temporal, and open-domain. The dataset originally included an adversarial question category, which was designed to test systems’ ability to recognize

<sup>1</sup><https://neo4j.com/>



unanswerable questions. However, this category was excluded from our evaluation because ground truth answers were unavailable, and the expected behavior for this question type is that the agent should recognize them as unanswerable.

### 3.2. Evaluation Metrics

Our evaluation framework implements a comprehensive approach to assess long-term memory capabilities in dialogue systems, considering both response quality and operational efficiency. We categorize our metrics into two distinct groups that together provide a holistic understanding of system performance.

**(1) Performance Metrics** Previous research in conversational AI (Goswami, 2025, Soni et al., 2024, Singh et al., 2020) has predominantly relied on **lexical similarity metrics** such as **F1 Score** ( $F_1$ ) and **BLEU-1** ( $B_1$ ). However, these metrics exhibit significant limitations when evaluating factual accuracy in conversational contexts. Consider a scenario where the ground truth answer is ‘Alice was born in March’ and a system generates ‘Alice is born in July.’ Despite containing a critical factual error regarding the birth month, traditional metrics would assign relatively high scores due to lexical overlap in the remaining tokens (‘Alice,’ ‘born,’ etc.). This fundamental limitation can lead to misleading evaluations that fail to capture semantic correctness. To address these shortcomings, we use **LLM-as-a-Judge** ( $J$ ) as a complementary evaluation metric. This approach leverages a separate, more capable LLM to assess response quality across multiple dimensions, including factual accuracy, relevance, completeness, and contextual appropriateness. The judge model analyzes the question, ground truth answer and the generated answer, providing a more nuanced evaluation that aligns better with human judgment. Due to the stochastic nature of  $J$  evaluations, we conducted 10 independent runs for each method on the entire dataset and report the mean scores along with  $\pm 1$  standard deviation. More details about the  $J$  is present in Appendix A.

**(2) Deployment Metrics** Beyond response quality, practical deployment considerations are crucial for real-world applications of long-term memory in AI agents. We systematically track **Token Consumption**, using ‘c1100k\_base’ encoding from `tiktoken`, measuring the number of tokens extracted during retrieval that serve as context for answering queries. For our memory-based models, these tokens represent the memories retrieved from the knowledge base, while for RAG-based models, they correspond to the total number of tokens in the retrieved text chunks. This distinction is important as it directly affects operational costs and system efficiency—whether processing concise memory facts or larger raw text segments. We further monitor **Latency**, (i) *search latency*: which captures the total time required to search the memory (in memory-based solutions) or chunk (in RAG-based solutions) and (ii) *total latency*: time to generate appropriate responses, consisting of both retrieval time (accessing memories or chunks) and answer generation time using the LLM.

The relationship between these metrics reveals important trade-offs in system design. For instance, more sophisticated memory architectures might achieve higher factual accuracy but at the cost of increased token consumption and latency. Our multi-dimensional evaluation methodology enables researchers and practitioners to make informed decisions based on their specific requirements, whether prioritizing response quality for critical applications or computational efficiency for real-time deployment scenarios.

### 3.3. Baselines

To comprehensively evaluate our approach, we compare against six distinct categories of baselines that represent the current state of conversational memory systems. These diverse baselines collectively provide a



robust framework for evaluating the effectiveness of different memory architectures across various dimensions, including factual accuracy, computational efficiency, and scalability to extended conversations. Where applicable, unless otherwise specified, we set the temperature to 0 to ensure the runs are as reproducible as possible.

**Established LOCOMO Benchmarks** We first establish a comparative foundation by evaluating previously benchmarked methods on the LOCOMO dataset. These include five established approaches: LoCoMo (Maharana et al., 2024), ReadAgent (Lee et al., 2024), MemoryBank (Zhong et al., 2024), MemGPT (Packer et al., 2023), and A-Mem (Xu et al., 2025). These established benchmarks not only provide direct comparison points with published results but also represent the evolution of conversational memory architectures across different algorithmic paradigms. For our evaluation, we select the metrics where gpt-4o-mini was used for the evaluation. More details about these benchmarks are mentioned in Appendix C.

**Open-Source Memory Solutions** Our second category consists of promising open-source memory architectures such as LangMem<sup>2</sup> (Hot Path) that have demonstrated effectiveness in related conversational tasks but have not yet been evaluated on the LOCOMO dataset. By adapting these systems to our evaluation framework, we broaden the comparative landscape and identify potential alternative approaches that may offer competitive performance. We initialized the LLM with gpt-4o-mini and used text-embedding-small-3 as the embedding model.

**Retrieval-Augmented Generation (RAG)** As a baseline, we treat the entire conversation history as a document collection and apply a standard RAG pipeline. We first segment each conversation into fixed-length chunks (128, 256, 512, 1024, 2048, 4096, and 8192 tokens), where 8192 is the maximum chunk size supported by our embedding model. All chunks are embedded using OpenAI’s text-embedding-small-3 to ensure consistent vector quality across configurations. At query time, we retrieve the top  $k$  chunks by semantic similarity and concatenate them as context for answer generation. Throughout our experiments we set  $k \in \{1, 2\}$ : with  $k=1$  only the single most relevant chunk is used, and with  $k=2$  the two most relevant chunks (up to 16384 tokens) are concatenated. We avoid  $k > 2$  since the average conversation length (26000 tokens) would be fully covered, negating the benefits of selective retrieval. By varying chunk size and  $k$ , we systematically evaluate RAG performance on long-term conversational memory tasks.

**Full-Context Processing** We adopt a straightforward approach by passing the entire conversation history within the context window of the LLM. This method leverages the model’s inherent ability to process sequential information without additional architectural components. While conceptually simple, this approach faces practical limitations as conversation length increases, eventually increasing token cost and latency. Nevertheless, it establishes an important reference point for understanding the value of more sophisticated memory mechanisms compared to direct processing of available context.

**Proprietary Models** We evaluate OpenAI’s memory<sup>3</sup> feature available in their ChatGPT interface, specifically using gpt-4o-mini for consistency. We ingest entire LOCOMO conversations with a prompt (see Appendix A) into single chat sessions, prompting memory generation with timestamps, participant names, and conversation

<sup>2</sup><https://langchain-ai.github.io/langmem/>

<sup>3</sup><https://openai.com/index/memory-and-new-controls-for-chatgpt/>



**Table 1:** Performance comparison of memory-enabled systems across different question types in the LOCOMO dataset. Evaluation metrics include F1 score ( $F_1$ ), BLEU-1 ( $B_1$ ), and LLM-as-a-Judge score ( $J$ ), with higher values indicating better performance. A-Mem\* represents results from our re-run of A-Mem to generate LLM-as-a-Judge scores by setting temperature as 0. Mem0<sup>g</sup> indicates our proposed architecture enhanced with graph memory. **Bold** denotes the best performance for each metric across all methods. (↑) represents higher score is better.

Method	Single Hop			Multi-Hop			Open Domain			Temporal		
	$F_1 \uparrow$	$B_1 \uparrow$	$J \uparrow$	$F_1 \uparrow$	$B_1 \uparrow$	$J \uparrow$	$F_1 \uparrow$	$B_1 \uparrow$	$J \uparrow$	$F_1 \uparrow$	$B_1 \uparrow$	$J \uparrow$
LoCoMo	25.02	19.75	–	12.04	11.16	–	40.36	29.05	–	18.41	14.77	–
ReadAgent	9.15	6.48	–	5.31	5.12	–	9.67	7.66	–	12.60	8.87	–
MemoryBank	5.00	4.77	–	5.56	5.94	–	6.61	5.16	–	9.68	6.99	–
MemGPT	26.65	17.72	–	9.15	7.44	–	41.04	34.34	–	25.52	19.44	–
A-Mem	27.02	20.09	–	12.14	12.00	–	44.65	37.06	–	45.85	36.67	–
A-Mem*	20.76	14.90	$39.79 \pm 0.38$	9.22	8.81	$18.85 \pm 0.31$	33.34	27.58	$54.05 \pm 0.22$	35.40	31.08	$49.91 \pm 0.31$
LangMem	35.51	26.86	$62.23 \pm 0.75$	26.04	<b>22.32</b>	$47.92 \pm 0.47$	40.91	33.63	$71.12 \pm 0.20$	30.75	25.84	$23.43 \pm 0.39$
Zep	35.74	23.30	$61.70 \pm 0.32$	19.37	14.82	$41.35 \pm 0.48$	<b>49.56</b>	38.92	<b>76.60 \pm 0.13</b>	42.00	34.53	$49.31 \pm 0.50$
OpenAI	34.30	23.72	$63.79 \pm 0.46$	20.09	15.42	$42.92 \pm 0.63$	39.31	31.16	$62.29 \pm 0.12$	14.04	11.25	$21.71 \pm 0.20$
Mem0	<b>38.72</b>	<b>27.13</b>	<b><math>67.13 \pm 0.65</math></b>	<b>28.64</b>	21.58	<b><math>51.15 \pm 0.31</math></b>	47.65	38.72	$72.93 \pm 0.11$	48.93	<b>40.51</b>	$55.51 \pm 0.34$
Mem0 <sup>g</sup>	38.09	26.03	$65.71 \pm 0.45$	24.32	18.82	$47.19 \pm 0.67$	49.27	<b>40.30</b>	$75.71 \pm 0.21$	<b>51.55</b>	40.28	<b><math>58.13 \pm 0.44</math></b>

text. These generated memories are then used as complete context for answering questions about each conversation, intentionally granting the OpenAI approach privileged access to all memories rather than only question-relevant ones. This methodology accommodates the lack of external API access for selective memory retrieval in OpenAI’s system for benchmarking.

**Memory Providers** We incorporate Zep (Rasmussen et al., 2025), a memory management platform designed for AI agents. Using their platform version, we conduct systematic evaluations across the LOCOMO dataset, maintaining temporal fidelity by preserving timestamp information alongside conversational content. This temporal anchoring ensures that time-sensitive queries can be addressed through appropriately contextualized memory retrieval, particularly important for evaluating questions that require chronological awareness. This baseline represents an important commercial implementation of memory management specifically engineered for AI agents.

## 4. Evaluation Results, Analysis and Discussion.

### 4.1. Performance Comparison Across Memory-Enabled Systems

Table 1 reports  $F_1$ ,  $B_1$  and  $J$  scores for our two architectures—Mem0 and Mem0<sup>g</sup>—against a suite of competitive baselines, as mentioned in Section 3, on single-hop, multi-hop, open-domain, and temporal questions. Overall, both of our models set new state-of-the-art marks in all the three evaluation metrics for most question types.

**Single-Hop Question Performance** Single-hop queries involve locating a single factual span contained within one dialogue turn. Leveraging its dense memories in natural language text, Mem0 secures the strongest results:  $F_1 = 38.72$ ,  $B_1 = 27.13$ , and  $J = 67.13$ . Augmenting the natural language memories with graph memory (Mem0<sup>g</sup>) yields marginal performance drop compared to Mem0, indicating that relational structure provides limited utility when the retrieval target occupies a single turn. Among the existing baselines, the full-context OpenAI run attains the next-best  $J$  score, reflecting the benefits of retaining the entire conversation in context, while LangMem and Zep both score around 8% relatively less against our models on  $J$  score. Previous LOCOMO



benchmarks such as A-mem lag by more than 25 points in J, underscoring the necessity of fine-grained, structured memory indexing even for simple retrieval tasks.

**Multi-Hop Question Performance** Multi-hop queries require synthesizing information dispersed across multiple conversation sessions, posing significant challenges in memory integration and retrieval. Mem0 clearly outperforms other methods with an F<sub>1</sub> score of 28.64 and a J score of 51.15, reflecting its capability to efficiently retrieve and integrate disparate information stored across sessions. Interestingly, the addition of graph memory in Mem0<sup>g</sup> does not provide performance gains here, indicating potential inefficiencies or redundancies in structured graph representations for complex integrative tasks compared to dense natural language memory alone. Baselines like LangMem show competitive performances, but their scores substantially trail those of Mem0, emphasizing the advantage of our refined memory indexing and retrieval mechanisms for complex query processing.

**Open-Domain Performance** In open-domain settings, the baseline Zep achieves the highest F<sub>1</sub> (49.56) and J (76.60) scores, edging out our methods by a narrow margin. In particular, Zep’s J score of 76.60 surpasses Mem0<sup>g</sup>’s 75.71 by just 0.89 percentage points and outperforms Mem0’s 72.93 by 3.67 points, highlighting a consistent, if slight, advantage in integrating conversational memory with external knowledge. Mem0<sup>g</sup> remains a strong runner-up, with a J of 75.71 reflecting high factual retrieval precision, while Mem0 follows with 72.93, demonstrating robust coherence. These results underscore that although structured relational memories (as in Mem0 and Mem0<sup>g</sup>) substantially improve open-domain retrieval, Zep maintains a small but meaningful lead.

**Temporal Reasoning Performance** Temporal reasoning tasks hinge on accurate modeling of event sequences, their relative ordering, and durations within conversational history. Our architectures demonstrate substantial improvements across all metrics, with Mem0<sup>g</sup> achieving the highest F<sub>1</sub>(51.55) and J (58.13), suggesting that structured relational representations in addition to natural language memories significantly aid in temporally grounded judgments. Notably, the base variant, Mem0, also provide a decent J score (55.51), suggesting that natural language alone can aid in temporally grounded judgments. Among baselines, OpenAI notably underperforms, with scores below 15%, primarily due to missing timestamps in most generated memories despite explicit prompting in the OpenAI ChatGPT to extract memories with timestamps. Other baselines such as A-Mem achieve respectable results, yet our models clearly advance the state-of-the-art, emphasizing the critical advantage of accurately leveraging both natural language contextualization and structured graph representations for temporal reasoning.

## 4.2. Cross-Category Analysis

The comprehensive evaluation across diverse question categories reveals that our proposed architectures, Mem0 and Mem0<sup>g</sup>, consistently achieve superior performance compared to baseline systems. For single-hop queries, Mem0 demonstrates particularly strong performance, benefiting from its efficient dense natural language memory structure. Although graph-based representations in Mem0<sup>g</sup> slightly lag behind in lexical overlap metrics for these simpler queries, they significantly enhance semantic coherence, as demonstrated by competitive J scores. This indicates that graph structures are more beneficial in scenarios involving nuanced relational context rather than straightforward retrieval. For multi-hop questions, Mem0 exhibits clear advantages by effectively synthesizing dispersed information across multiple sessions, confirming that natural language memories provide sufficient representational richness for these integrative tasks. Surprisingly, the

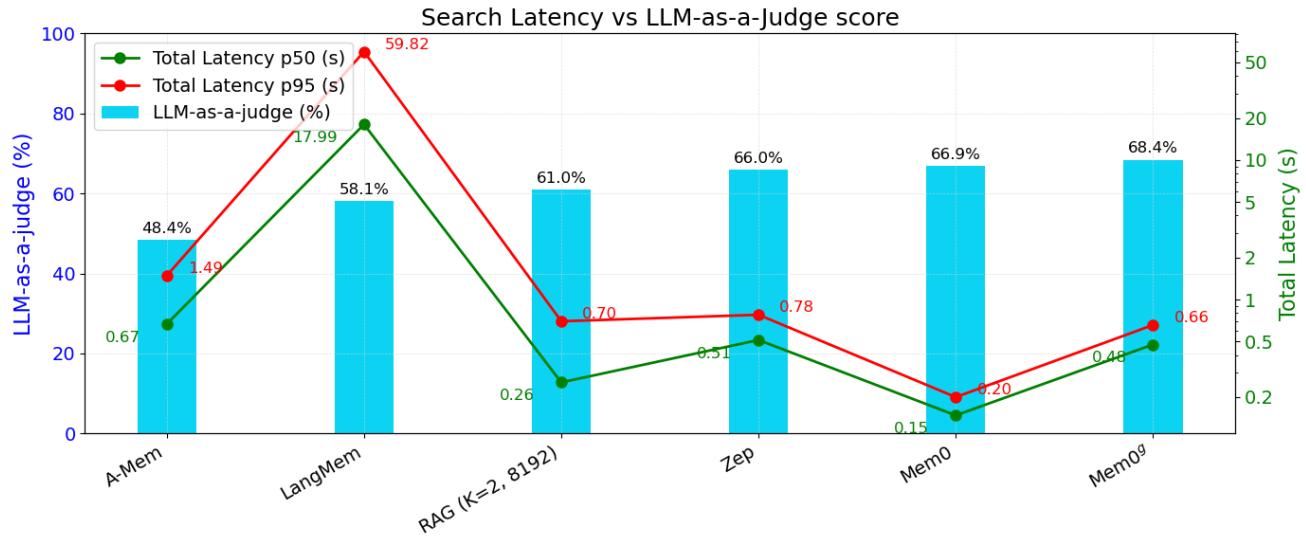


**Table 2:** Performance comparison of various baselines with proposed methods. Latency measurements show p50 (median) and p95 (95th percentile) values in seconds for both search time (time taken to fetch memories/chunks) and total time (time to generate the complete response). Overall LLM-as-a-Judge score (J) represents the quality metric of the generated responses on the entire LOCOMO dataset.

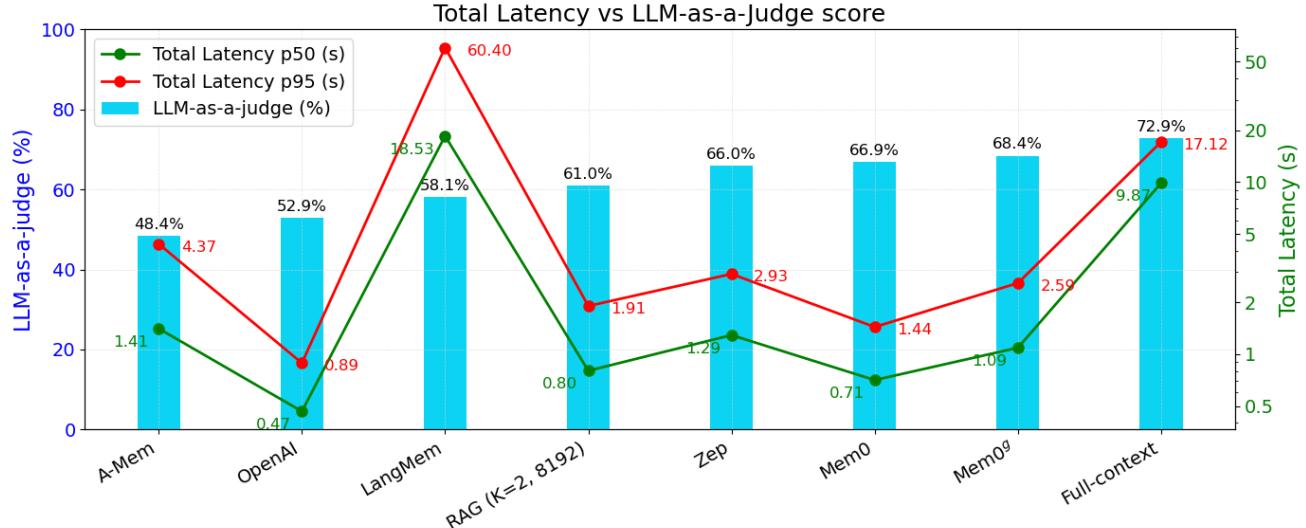
Method	K	chunk size / memory tokens	Latency (seconds)				Overall J	
			Search		Total			
			p50	p95	p50	p95		
RAG	1	128	0.281	0.823	0.774	1.825	$47.77 \pm 0.23\%$	
		256	0.251	0.710	0.745	1.628	$50.15 \pm 0.16\%$	
		512	0.240	0.639	0.772	1.710	$46.05 \pm 0.14\%$	
		1024	0.240	0.723	0.821	1.957	$40.74 \pm 0.17\%$	
		2048	0.255	0.752	0.996	2.182	$37.93 \pm 0.12\%$	
		4096	0.254	0.719	1.093	2.711	$36.84 \pm 0.17\%$	
		8192	0.279	0.838	1.396	4.416	$44.53 \pm 0.13\%$	
		128	0.267	0.624	0.766	1.829	$59.56 \pm 0.19\%$	
RAG	2	256	0.255	0.699	0.802	1.907	$60.97 \pm 0.20\%$	
		512	0.247	0.746	0.829	1.729	$58.19 \pm 0.18\%$	
		1024	0.238	0.702	0.860	1.850	$50.68 \pm 0.13\%$	
		2048	0.261	0.829	1.101	2.791	$48.57 \pm 0.22\%$	
		4096	0.266	0.944	1.451	4.822	$51.79 \pm 0.15\%$	
		8192	0.288	1.124	2.312	9.942	$60.53 \pm 0.16\%$	
Full-context		26031	-	-	9.870	17.117	<b><math>72.90 \pm 0.19\%</math></b>	
A-Mem		2520	0.668	1.485	1.410	4.374	$48.38 \pm 0.15\%$	
LangMem		127	17.99	59.82	18.53	60.40	$58.10 \pm 0.21\%$	
Zep		3911	0.513	0.778	1.292	2.926	$65.99 \pm 0.16\%$	
OpenAI		4437	-	-	0.466	0.889	$52.90 \pm 0.14\%$	
Mem0		1764	<b>0.148</b>	<b>0.200</b>	<b>0.708</b>	<b>1.440</b>	$66.88 \pm 0.15\%$	
Mem0 <sup>g</sup>		3616	0.476	0.657	1.091	2.590	$68.44 \pm 0.17\%$	

expected relational advantages of Mem0<sup>g</sup> do not translate into better outcomes here, suggesting potential overhead or redundancy when navigating more intricate graph structures in multi-step reasoning scenarios.

In temporal reasoning, Mem0<sup>g</sup> substantially outperforms other methods, validating that structured relational graphs excel in capturing chronological relationships and event sequences. The presence of explicit relational context significantly enhances Mem0<sup>g</sup>'s temporal coherence, outperforming Mem0's dense memory storage and highlighting the importance of precise relational representations when tracking temporally sensitive information. Open-domain performance further reinforces the value of relational modeling. Mem0<sup>g</sup>, benefiting from the relational clarity of graph-based memory, closely competes with the top-performing baseline (Zep). This competitive result underscores Mem0<sup>g</sup>'s robustness in integrating external knowledge through relational clarity, suggesting an optimal synergy between structured memory and open-domain information synthesis.



(a) Comparison of *search* latency at p50 (median) and p95 (95th percentile) across different memory methods (Mem0, Mem0<sup>8</sup>, best RAG variant, Zep, LangMem, and A-Mem). The bar heights represent J scores (left axis), while the line plots show search latency in seconds (right axis scaled in log).



(b) Comparison of *total* response latency at p50 and p95 across different memory methods (Mem0, Mem0<sup>8</sup>, best RAG variant, Zep, LangMem, OpenAI, full-context, and A-Mem). The bar heights represent J scores (left axis), and the line plots capture end-to-end latency in seconds (right axis scaled in log).

**Figure 4: Latency Analysis of Different Memory Approaches.** These subfigures illustrate the J scores and latency comparison of various selected methods from Table 2. Subfigure (a) highlights the *search/retrieval* latency prior to answer generation, while Subfigure (b) shows the *total* latency (including LLM inference). Both plots overlay each method’s J score for a holistic view of their accuracy and efficiency.

Overall, our analysis indicates complementary strengths of Mem0 and Mem0<sup>8</sup> across various task demands: dense, natural-language-based memory offers significant efficiency for simpler queries, while explicit relational modeling becomes essential for tasks demanding nuanced temporal and contextual integration. These findings reinforce the importance of adaptable memory structures tailored to specific reasoning contexts in AI agent deployments.



### 4.3. Performance Comparison of Mem0 and Mem0<sup>g</sup> Against RAG Approaches and Full-Context Model

Comparisons in Table 2, focusing on the ‘Overall J’ column, reveal that both Mem0 and Mem0<sup>g</sup> consistently outperform all RAG configurations, which vary chunk sizes (128–8192 tokens) and retrieve either one ( $k=1$ ) or two ( $k=2$ ) chunks. Even the strongest RAG approach peaks at around 61% in the J metric, whereas Mem0 reaches 67%—about a 10% relative improvement—and Mem0<sup>g</sup> reaches over 68%, achieving around a 12% relative gain. These advances underscore the advantage of capturing only the most salient facts in memory, rather than retrieving large chunk of original text. By converting the conversation history into concise, structured representations, Mem0 and Mem0<sup>g</sup> mitigate noise and surface more precise cues to the LLM, leading to better answers as evaluated by an external LLM (J).

Despite these improvements, a full-context method that ingests a chunk of roughly 26,000 tokens still achieves the highest J score (approximately 73%). However, as shown in Figure 4b, it also incurs a very high total p95 latency—around 17 seconds—since the model must read the entire conversation on every query. By contrast, Mem0 and Mem0<sup>g</sup> significantly reduce token usage and thus achieve lower p95 latencies of around 1.44 seconds (a 92% reduction) and 2.6 seconds (a 85% reduction), respectively over full-context approach. Although the full-context approach can provide a slight accuracy edge, the memory-based systems offer a more practical trade-off, maintaining near-competitive quality while imposing only a fraction of the token and latency cost. As conversation length increases, full-context approaches suffer from exponential growth in computational overhead (evident in Table 2 where total p95 latency increases significantly with larger  $k$  values or chunk sizes). This increase in input chunks leads to longer response times and higher token consumption costs. In contrast, memory-focused approaches like Mem0 and Mem0<sup>g</sup> maintain consistent performance regardless of conversation length, making them substantially more viable for production-scale deployments where efficiency and responsiveness are critical.

### 4.4. Latency Analysis

Table 2 provides a comprehensive performance comparison of various retrieval and memory methodologies, presenting median (p50) and tail (p95) latencies for both the search phase and total response generation across the LOCOMO dataset. Our analysis reveals distinct performance patterns governed by architectural choices. Memory-centric architectures demonstrate different performance characteristics. A-Mem, despite its larger memory store, incurs substantial search overhead (p50: 0.668s), resulting in total median latencies of 1.410s. LangMem exhibits even higher search latencies (p50: 17.99s, p95: 59.82s), rendering it impractical for interactive applications. Zep achieves moderate performance (p50 total: 1.292s). The full-context baseline, which processes the entire conversation history without retrieval, fundamentally differs from retrieval-based approaches. By passing the entire conversation context (26000 tokens) directly to the LLM, it eliminates search overhead but incurs extreme total latencies (p50: 9.870s, p95: 17.117s). Similarly, the OpenAI implementation does not perform memory search, as it processes manually extracted memories from their playground. While this approach achieves impressive response generation times (p50: 0.466s, p95: 0.889s), it requires pre-extraction of relevant context, which is not reflected in the reported metrics.

Our proposed Mem0 approach achieves the lowest search latency among all methods (p50: 0.148s, p95: 0.200s) as illustrated in Figure 4a. This efficiency stems from our selective memory retrieval mechanism and infra improvements that dynamically identifies and retrieves only the most salient information rather than fixed-size chunks. Consequently, Mem0 maintains the lowest total median latency (0.708s) with remarkably contained p95 values (1.440s), making it particularly suitable for latency-sensitive applications such as interactive AI agents. The graph-enhanced Mem0<sup>g</sup> variant introduces additional relational modeling capabilities at a moderate latency cost, with search times (0.476s) still outperforming all existing memory



solutions and baselines. Despite this increase, Mem0<sup>g</sup> maintains competitive total latencies (p50: 1.091s, p95: 2.590s) while achieving the highest J score (68.44%) across all methods—trailing only the computationally prohibitive full-context approach. This performance profile demonstrates our methods’ ability to balance response quality and computational efficiency, offering a compelling solution for production AI agents where both factors are critical constraints.

→ interessant  
re TFG

#### 4.5. Memory System Overhead: Token Analysis and Construction Time

We measure the average token budget required to materialise each system’s long-term memory store. Mem0 encodes complete dialogue turns in a natural language representation and therefore occupies only **7k** tokens per conversation on an average. Whereas Mem0<sup>g</sup> roughly doubles the footprint to **14k** tokens, due to the introduction of graph memories which includes nodes and corresponding relationships. In stark contrast, Zep’s memory graph consumes in excess of **600k** tokens. The inflation arises from Zep’s design choice to cache a full abstractive summary at every node while also storing facts on the connecting edges, leading to extensive redundancy across the graph. For perspective, supplying the *entire* raw conversation context to the language model—without any memory abstraction—amounts to roughly **26k** tokens on average, 20 times less relative to Zep’s graph. Beyond token inefficiency, our experiments revealed significant operational bottlenecks with Zep. After adding memories to Zep’s system, we observed that immediate memory retrieval attempts often failed to answer our queries correctly. Interestingly, re-running identical searches after a delay of several hours yielded considerably better results. This latency suggests that Zep’s graph construction involves multiple asynchronous LLM calls and extensive background processing, making the memory system impractical for real-time applications. In contrast, Mem0 graph construction completes in under a minute even in worst-case scenarios, allowing users to immediately leverage newly added memories for query responses.

These findings highlight that Zep not only replicates identical knowledge fragments across multiple nodes, but also introduces significant operational delays. Our architectures—Mem0 and Mem0<sup>g</sup>—preserve the same information at a fraction of the token cost and with substantially faster memory availability, offering a more memory-efficient and operationally responsive representation.

## 5. Conclusion and Future Work

We have introduced Mem0 and Mem0<sup>g</sup>, two complementary memory architectures that overcome the intrinsic limitations of fixed context windows in LLMs. By dynamically extracting, consolidating, and retrieving compact memory representations, Mem0 achieves state-of-the-art performance across single-hop and multi-hop reasoning, while Mem0<sup>g</sup>’s graph-based extensions unlock significant gains in temporal and open-domain tasks. On the LOCOMO benchmark, our methods deliver 5%, 11%, and 7% relative improvements in single-hop, temporal, and multi-hop reasoning question types over best performing methods in respective question type and reduce p95 latency by over 91% compared to full-context baselines—demonstrating a powerful balance between precision and responsiveness. Mem0’s dense memory pipeline excels at rapid retrieval for straightforward queries, minimizing token usage and computational overhead. In contrast, Mem0<sup>g</sup>’s structured graph representations provide nuanced relational clarity, enabling complex event sequencing and rich context integration without sacrificing practical efficiency. Together, they form a versatile memory toolkit that adapts to diverse conversational demands while remaining deployable at scale.

Future research directions include optimizing graph operations to reduce the latency overhead in Mem0<sup>g</sup>, exploring hierarchical memory architectures that blend efficiency with relational representation, and developing more sophisticated memory consolidation mechanisms inspired by human cognitive processes.



Additionally, extending our memory frameworks to domains beyond conversational scenarios, such as procedural reasoning and multimodal interactions, would further validate their broader applicability. By addressing the fundamental limitations of fixed context windows, our work represents a significant advancement toward conversational AI systems capable of maintaining coherent, contextually rich interactions over extended periods, much like their human counterparts.

## 6. Acknowledgments

We would like to express our sincere gratitude to Harsh Agarwal, Shyamal Anadkat, Prithvijit Chattopadhyay, Siddesh Choudhary, Rishabh Jain, and Vaibhav Pandey for their invaluable insights and thorough reviews of early drafts. Their constructive comments and detailed suggestions helped refine the manuscript, enhancing both its clarity and overall quality. We deeply appreciate their generosity in dedicating time and expertise to this work.

## References

- Anthropic. Model card and evaluations for claude models. Technical report, Anthropic, February 2025. URL <https://www.anthropic.com/news/clause-3-7-sonnet>.
- Jan Assmann. Communicative and cultural memory. In *Cultural memories: The geographical point of view*, pages 15–27. Springer, 2011.
- Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. Recurrent memory transformer. *Advances in Neural Information Processing Systems*, 35:11079–11091, 2022.
- Prateek Chhikara, Jiarui Zhang, Filip Ilievski, Jonathan Francis, and Kaixin Ma. Knowledge-enhanced agents for interactive text games. In *Proceedings of the 12th Knowledge Capture Conference 2023*, pages 157–165, 2023.
- Fergus IM Craik and Janine M Jennings. Human memory. 1992.
- Gaurav Goswami. Dissecting the metrics: How different evaluation approaches yield diverse results for conversational ai. *Authorea Preprints*, 2025.
- Tianyu Guo, Druv Pai, Yu Bai, Jiantao Jiao, Michael Jordan, and Song Mei. Active-dormant attention heads: Mechanistically demystifying extreme-token phenomena in llms. In *NeurIPS 2024 Workshop on Mathematics of Modern Machine Learning*, 2024.
- Kostas Hatalis, Despina Christou, Joshua Myers, Steven Jones, Keith Lambert, Adam Amos-Binks, Zohreh Dannenhauer, and Dustin Dannenhauer. Memory matters: The need to improve long-term memory in llm-agents. In *Proceedings of the AAAI Symposium Series*, volume 2, pages 277–280, 2023.
- Zihong He, Weizhe Lin, Hao Zheng, Fan Zhang, Matt W Jones, Laurence Aitchison, Xuhai Xu, Miao Liu, Per Ola Kristensson, and Junxiao Shen. Human-inspired perspectives: A survey on ai long-term memory. *arXiv preprint arXiv:2411.00489*, 2024.
- Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.



Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Kuang-Huei Lee, Xinyun Chen, Hiroki Furuta, John Canny, and Ian Fischer. A human-inspired reading agent with gist memory of very long contexts. In *International Conference on Machine Learning*, pages 26396–26415. PMLR, 2024.

Lei Liu, Xiaoyan Yang, Yue Shen, Binbin Hu, Zhiqiang Zhang, Jinjie Gu, and Guannan Zhang. Think-in-memory: Recalling and post-thinking enable llms with long-term memory. *arXiv preprint arXiv:2311.08719*, 2023.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13851–13870, 2024.

Bodhisattwa Prasad Majumder, Bhavana Dalvi Mishra, Peter Jansen, Oyvind Tafjord, Niket Tandon, Li Zhang, Chris Callison-Burch, and Peter Clark. Clin: A continually learning language agent for rapid task adaptation and generalization. In *First Conference on Language Modeling*.

Elliot Nelson, Georgios Kollias, Payel Das, Subhajit Chaudhury, and Soham Dan. Needle in the haystack for memory based large language models. *arXiv preprint arXiv:2407.01437*, 2024.

Charles Packer, Vivian Fang, Shishir\_G Patil, Kevin Lin, Sarah Wooders, and Joseph\_E Gonzalez. Memgpt: Towards llms as operating systems. 2023.

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

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

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

Prabhjot Singh, Prateek Chhikara, and Jasmeet Singh. An ensemble approach for extractive text summarization. In *2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE)*, pages 1–7. IEEE, 2020.

Arpita Soni, Rajeev Arora, Anoop Kumar, and Dheerendra Panwar. Evaluating domain coverage in low-resource generative chatbots: A comparative study of open-domain and closed-domain approaches using bleu scores. In *2024 International Conference on Electrical Electronics and Computing Technologies (ICEECT)*, volume 1, pages 1–6. IEEE, 2024.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

Joan C Timoneda and Sebastián Vallejo Vera. Memory is all you need: Testing how model memory affects llm performance in annotation tasks. *arXiv preprint arXiv:2503.04874*, 2025.



- Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*, 2025.
- Yangyang Yu, Haohang Li, Zhi Chen, Yuechen Jiang, Yang Li, Denghui Zhang, Rong Liu, Jordan W Suchow, and Khaldoun Khashanah. Finmem: A performance-enhanced llm trading agent with layered memory and character design. In *Proceedings of the AAAI Symposium Series*, volume 3, pages 595–597, 2024.
- Jiarui Zhang. Guided profile generation improves personalization with large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 4005–4016, 2024.
- Jiarui Zhang, Filip Ilievski, Kaixin Ma, Jonathan Francis, and Alessandro Oltramari. A study of zero-shot adaptation with commonsense knowledge. In *AKBC*, 2022.
- Zeyu Zhang, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Quanyu Dai, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. A survey on the memory mechanism of large language model based agents. *arXiv preprint arXiv:2404.13501*, 2024.
- Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19724–19731, 2024.



## Appendix

### A. Prompts

In developing our LLM-as-a-Judge prompt, we adapt elements from the prompt released by Packer et al. (2023).

#### PROMPT TEMPLATE FOR LLM AS A JUDGE

Your task is to label an answer to a question as "CORRECT" or "WRONG". You will be given the following data: (1) a question (posed by one user to another user), (2) a 'gold' (ground truth) answer, (3) a generated answer which you will score as CORRECT/WRON

The point of the question is to ask about something one user should know about the other user based on their prior conversations. The gold answer will usually be a concise and short answer that includes the referenced topic, for example:

Question: Do you remember what I got the last time I went to Hawaii?

Gold answer: A shell necklace

The generated answer might be much longer, but you should be generous with your grading - as long as it touches on the same topic as the gold answer, it should be counted as CORRECT.

For time related questions, the gold answer will be a specific date, month, year, etc. The generated answer might be much longer or use relative time references (like 'last Tuesday' or 'next month'), but you should be generous with your grading - as long as it refers to the same date or time period as the gold answer, it should be counted as CORRECT. Even if the format differs (e.g., 'May 7th' vs '7 May'), consider it CORRECT if it's the same date.

Now it's time for the real question:

Question: {question}

Gold answer: {gold\_answer}

Generated answer: {generated\_answer}

First, provide a short (one sentence) explanation of your reasoning, then finish with CORRECT or WRONG. Do NOT include both CORRECT and WRONG in your response, or it will break the evaluation script.

Just return the label CORRECT or WRONG in a json format with the key as "label".



## PROMPT TEMPLATE FOR RESULTS GENERATION (MEM0)

You are an intelligent memory assistant tasked with retrieving accurate information from conversation memories.

### # CONTEXT:

You have access to memories from two speakers in a conversation. These memories contain timestamped information that may be relevant to answering the question.

### # INSTRUCTIONS:

1. Carefully analyze all provided memories from both speakers
2. Pay special attention to the timestamps to determine the answer
3. If the question asks about a specific event or fact, look for direct evidence in the memories
4. If the memories contain contradictory information, prioritize the most recent memory
5. If there is a question about time references (like "last year", "two months ago", etc.), calculate the actual date based on the memory timestamp. For example, if a memory from 4 May 2022 mentions "went to India last year," then the trip occurred in 2021.
6. Always convert relative time references to specific dates, months, or years. For example, convert "last year" to "2022" or "two months ago" to "March 2023" based on the memory timestamp. Ignore the reference while answering the question.
7. Focus only on the content of the memories from both speakers. Do not confuse character names mentioned in memories with the actual users who created those memories.
8. The answer should be less than 5-6 words.

### # APPROACH (Think step by step):

1. First, examine all memories that contain information related to the question
2. Examine the timestamps and content of these memories carefully
3. Look for explicit mentions of dates, times, locations, or events that answer the question
4. If the answer requires calculation (e.g., converting relative time references), show your work
5. Formulate a precise, concise answer based solely on the evidence in the memories
6. Double-check that your answer directly addresses the question asked
7. Ensure your final answer is specific and avoids vague time references

Memories for user {speaker\_1\_user\_id}:  
{speaker\_1\_memories}

Memories for user {speaker\_2\_user\_id}:  
{speaker\_2\_memories}

Question: {question}

Answer:



### PROMPT TEMPLATE FOR RESULTS GENERATION (MEM0<sup>g</sup>)

*(same as previous)*

# APPROACH (Think step by step):

1. First, examine all memories that contain information related to the question
2. Examine the timestamps and content of these memories carefully
3. Look for explicit mentions of dates, times, locations, or events that answer the question
4. If the answer requires calculation (e.g., converting relative time references), show your work
5. Analyze the knowledge graph relations to understand the user's knowledge context
6. Formulate a precise, concise answer based solely on the evidence in the memories
7. Double-check that your answer directly addresses the question asked
8. Ensure your final answer is specific and avoids vague time references

Memories for user {speaker\_1\_user\_id}:  
{speaker\_1\_memories}

Relations for user {speaker\_1\_user\_id}:  
{speaker\_1\_graph\_memories}

Memories for user {speaker\_2\_user\_id}:  
{speaker\_2\_memories}

Relations for user {speaker\_2\_user\_id}:  
{speaker\_2\_graph\_memories}

Question: {question}

Answer:

### PROMPT TEMPLATE FOR OPENAI CHATGPT

Can you please extract relevant information from this conversation and create memory entries for each user mentioned? Please store these memories in your knowledge base in addition to the timestamp provided for future reference and personalized interactions.

(1:56 pm on 8 May, 2023) Caroline: Hey Mel! Good to see you! How have you been?  
(1:56 pm on 8 May, 2023) Melanie: Hey Caroline! Good to see you! I'm swamped with the kids & work. What's up with you? Anything new?  
(1:56 pm on 8 May, 2023) Caroline: I went to a LGBTQ support group yesterday and it was so powerful.  
...



## B. Algorithm

---

**Algorithm 1** Memory Management System: Update Operations
 

---

```

1: Input: Set of retrieved memories  $F$ , Existing memory store  $M = \{m_1, m_2, \dots, m_n\}$ 
2: Output: Updated memory store  $M'$ 
3: procedure UPDATEMEMORY( $F, M$ )
4:   for each fact  $f \in F$  do
5:      $operation \leftarrow \text{CLASSIFYOPERATION}(f, M)$             $\triangleright$  Execute appropriate operation based on
       classification
6:     if  $operation = \text{ADD}$  then
7:        $id \leftarrow \text{GenerateUniqueID}()$ 
8:        $M \leftarrow M \cup \{(id, f, "ADD")\}$                     $\triangleright$  Add new fact with unique identifier
9:     else if  $operation = \text{UPDATE}$  then
10:     $m_i \leftarrow \text{FindRelatedMemory}(f, M)$ 
11:    if  $\text{InformationContent}(f) > \text{InformationContent}(m_i)$  then
12:       $M \leftarrow (M \setminus \{m_i\}) \cup \{(id, f, "UPDATE")\}$        $\triangleright$  Replace with richer information
13:    end if
14:    else if  $operation = \text{DELETE}$  then
15:       $m_i \leftarrow \text{FindContradictedMemory}(f, M)$ 
16:       $M \leftarrow M \setminus \{m_i\}$                                  $\triangleright$  Remove contradicted information
17:    else if  $operation = \text{NOOP}$  then
18:      No operation performed                                $\triangleright$  Fact already exists or is irrelevant
19:    end if
20:  end for
21:  return  $M$ 
22: end procedure

23: function CLASSIFYOPERATION( $f, M$ )
24:   if  $\neg \text{SemanticallySimilar}(f, M)$  then
25:     return ADD                                      $\triangleright$  New information not present in memory
26:   else if  $\text{Contradicts}(f, M)$  then
27:     return DELETE                                 $\triangleright$  Information conflicts with existing memory
28:   else if  $\text{Augments}(f, M)$  then
29:     return UPDATE                                 $\triangleright$  Enhances existing information in memory
30:   else
31:     return NOOP                                  $\triangleright$  No change required
32:   end if
33: end function
  
```

---

## C. Selected Baselines

**LoCoMo** The LoCoMo framework implements a sophisticated memory pipeline that enables LLM agents to maintain coherent, long-term conversations. At its core, the system divides memory into short-term and long-term components. After each conversation session, agents generate summaries (stored as short-term memory) that distill key information from that interaction. Simultaneously, individual conversation turns are



transformed into ‘*observations*’ - factual statements about each speaker’s persona and life events that are stored in long-term memory with references to the specific dialog turns that produced them. When generating new responses, agents leverage both the most recent session summary and selectively retrieve relevant observations from their long-term memory. This dual-memory approach is further enhanced by incorporating a temporal event graph that tracks causally connected life events occurring between conversation sessions. By conditioning responses on retrieved memories, current conversation context, persona information, and intervening life events, the system enables agents to maintain consistent personalities and recall important details across conversations spanning hundreds of turns and dozens of sessions.

**ReadAgent** ReadAgent addresses the fundamental limitations of LLMs by emulating how humans process lengthy texts through a sophisticated three-stage pipeline. First, in Episode Pagination, the system intelligently segments text at natural cognitive boundaries rather than arbitrary cutoffs. Next, during Memory Gisting, it distills each segment into concise summaries that preserve essential meaning while drastically reducing token count—similar to how human memory retains the substance of information without verbatim recall. Finally, when tasked with answering questions, the Interactive Lookup mechanism examines these gists and strategically retrieves only the most relevant original text segments for detailed processing. This human-inspired approach enables LLMs to effectively manage documents up to 20 times longer than their normal context windows. By balancing global understanding through gists with selective attention to details, ReadAgent achieves both computational efficiency and improved comprehension, demonstrating that mimicking human cognitive processes can significantly enhance AI text processing capabilities.

**MemoryBank** The MemoryBank system enhances LLMs with long-term memory through a sophisticated three-part pipeline. At its core, the Memory Storage component warehouses detailed conversation logs, hierarchical event summaries, and evolving user personality profiles. When a new interaction occurs, the Memory Retrieval mechanism employs a dual-tower dense retrieval model to extract contextually relevant past information. The Memory Updating component, provides a human-like forgetting mechanism where memories strengthen when recalled and naturally decay over time if unused. This comprehensive approach enables AI companions to recall pertinent information, maintain contextual awareness across extended interactions, and develop increasingly accurate user portraits, resulting in more personalized and natural long-term conversations.

**MemGPT** The MemGPT system introduces an operating system-inspired approach to overcome the context window limitations inherent in LLMs. At its core, MemGPT employs a sophisticated memory management pipeline consisting of three key components: a hierarchical memory system, self-directed memory operations, and an event-based control flow mechanism. The system divides available memory into ‘*main context*’ (analogous to RAM in traditional operating systems) and ‘*external context*’ (analogous to disk storage). The main context—which is bound by the LLM’s context window—contains system instructions, recent conversation history, and working memory that can be modified by the model. The external context stores unlimited information outside the model’s immediate context window, including complete conversation histories and archival data. When the LLM needs information not present in main context, it can initiate function calls to search, retrieve, or modify content across these memory tiers, effectively ‘*paging*’ relevant information in and out of its limited context window. This OS-inspired architecture enables MemGPT to maintain conversational coherence over extended interactions, manage documents that exceed standard context limits, and perform multi-hop information retrieval tasks—all while operating with fixed-context



models. The system’s ability to intelligently manage its own memory resources provides the illusion of infinite context, significantly extending what’s possible with current LLM technology.

**A-Mem** The A-Mem model introduces an agentic memory system designed for LLM agents. This system dynamically structures and evolves memories through interconnected notes. Each note captures interactions enriched with structured attributes like keywords, contextual descriptions, and tags generated by the LLM. Upon creating a new memory, A-MEM uses semantic embeddings to retrieve relevant existing notes, then employs an LLM-driven approach to establish meaningful links based on similarities and shared attributes. Crucially, the memory evolution mechanism updates existing notes dynamically, refining their contextual information and attributes whenever new relevant memories are integrated. Thus, memory structure continually evolves, allowing richer and contextually deeper connections among memories. Retrieval from memory is conducted through semantic similarity, providing relevant historical context during agent interactions