MemForest: Efficient Agent Memory Management via EventTree Partitioning and Progressive Merging

Junxi Wang[1,2], Te Sun[1], Jiayi Zhu[1], Chen Zhang[4] Siyuan Li[5], Xuyang Liu[6], Zichen Wen[1,3], Xiaobing Tu[7], Jinkui Ren[7], Xiantao Zhang[7], Ziqi Yuan[8], Linfeng Zhang[1]†
[1]Shanghai Jiao Tong University, [2]Fudan University, [3]Shanghai AI Lab, [4]Nanjing University
[5]HIT, [6]The Hong Kong Polytechnic University, [7]Alibaba Group, [8]Tsinghua University
junxiwang182@gmail.com, zhanglinfeng@sjtu.edu.cn

Abstract

Agent memory systems have demonstrated significant potential in tasks such as long-term dialogue, personalized assistants, and video understanding. However, as inference progresses, continuously accumulated memory imposes substantial storage and retrieval burdens. To address this issue, we propose **MemForest**, a general memory compression framework adaptable to various agent memory systems. Specifically, MemForest leverages both global semantic similarity and local temporal continuity of memory events to partition the historical memory into a set of event-centric independent units. For each independent unit, the framework constructs a maximum spanning tree structure, referred to as an EventTree, and performs progressive merging by iteratively selecting high-weight edges, thereby effectively compressing redundant memory nodes and reducing storage overhead. In addition, we introduce an anchor-guided propagation retrieval mechanism, which retrieves more relevant memory nodes from the temporal neighborhoods of key memory nodes, thereby enabling more accurate memory retrieval. Extensive experiments demonstrate the effectiveness of MemForest. Under the unimodal Mem0 framework, across three benchmarks (LoCoMo, LongMemEval, and PersonaMem), MemForest preserves **97.1%** of the original performance while compressing **50%** of historical memory, achieving a **1.89×** retrieval speedup. Under the multimodal M3-Agent framework, across two benchmarks (M3-Bench-robot and M3-Bench-web), MemForest retains **99.7%** of the original performance under a **50%** compression ratio, while achieving a **2.24×** retrieval speedup. *Our code is available at https://github.com/Celina-love-sweet/MemForest.*

1 Introduction

In recent years, the application of LLMs has grown rapidly, expanding from dialogue tasks [3, 36] to a range of problem-solving scenarios [6, 40, 20, 46, 59, 60, 21, 22, 53, 13]. In these contexts, long-term memory is crucial for maintaining contextual continuity and capturing dependencies across time. To address this, agent memory systems have emerged, storing historical interactions and key information as external memory to enhance a model's ability to capture long-term dependencies. These systems have shown significant potential in long-term dialogue [8, 14, 24], personalized assistants [32, 37, 42], and video understanding [27, 51]. However, as dialogue turns accumulate, memory growth poses challenges to both storage and retrieval efficiency. As shown in Fig. 1(a), under the Mem0 framework [8], expanding memory nodes increases storage pressure and reduces retrieval efficiency. Therefore, exploring efficient memory compression methods has become an important and urgent problem.

†Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

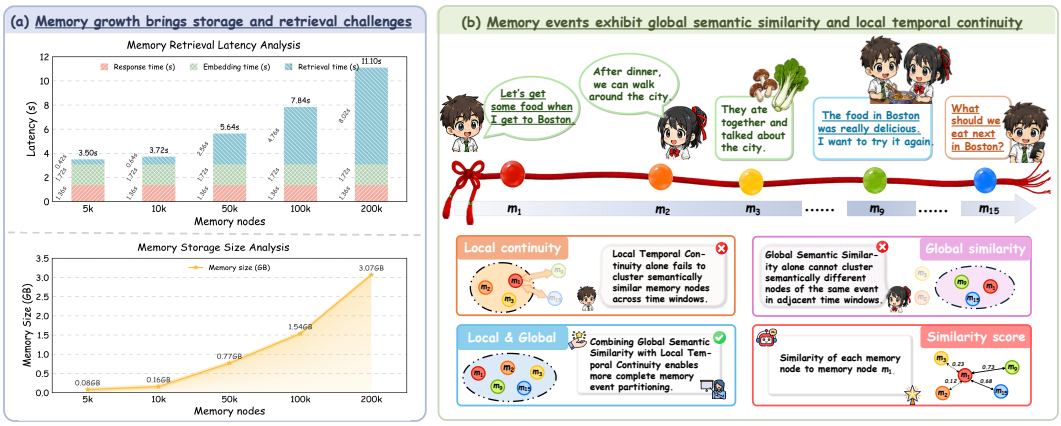


Figure 1: **(a) Challenges brought by the continuous growth of memory.** Each complete user Q&A involves three time costs: *Embedding time:* converting the query into a vector; *Retrieval time:* searching for corresponding memory nodes via similarity computation; *Response time:* generating the model's answer. **(b) An example from the LoCoMo benchmark.** This example illustrates that memory events exhibit *global semantic similarity and local temporal continuity*.

However, existing research on memory compression largely focuses on the generation stage [12, 26], which cannot fundamentally curb the continuous growth of memory, as stored memories still accumulate over time. Therefore, it is necessary to adopt a post-processing perspective and perform compression on already generated memories to systematically optimize storage efficiency and retrieval performance. Although recent work has begun to explore the feasibility of post-processing memory compression [41], this method typically relies on graph-structured memory representations and is restricted to streaming video scenarios, limiting its general applicability. Consequently, designing a general and scalable memory compression framework remains an important and open challenge.

Since stored memories are retrieved as external context, memory compression is closely related to context-length reduction. Recent token compression methods [5, 7, 44, 50, 52, 56, 57] reduce context via similarity-based merging or diversity-based pruning, mainly for MLLMs. However, unlike redundant visual inputs, textual memories are more semantically rich and information-dense [29, 55, 57], making compression challenging; existing methods also overlook memory-specific properties such as event differentiation and temporal relationships. Inspired by the event-level organization of human memory [23, 39], we partition historical memory into events and identify two key features for each memory event:

(I) Global Semantic Similarity: Fig. 1(b) suggests that semantically similar memory nodes may recur across distant temporal windows. For example, the similarity between memory node m_1 and the temporally distant memory node m_9 is **0.73**, and that between node m_1 and node m_{15} is **0.68**. This indicates that ***similar memory nodes within the same event can reappear over long time spans***.

(II) Local Temporal Continuity: Fig. 1(b) suggests that within adjacent temporal windows, memory nodes may exhibit substantial semantic differences yet still belong to the same event. For example, the similarity between memory node m_1 and node m_2 is only **0.12**, and that between node m_1 and node m_3 is only **0.23**, yet all three pertain to "Boston dining plan". This indicates that ***even when semantic similarity is low, temporally proximate memory nodes may still belong to the same memory event***.

Consequently, the above analysis indicates that when performing memory compression, it is necessary to consider both global semantic similarity and local temporal continuity. To this end, we propose **MemForest**, a general agent memory compression framework adaptable to various agent memory systems. Specifically, MemForest leverages the **EventTree Semantic-Temporal Partitioning** module to combine global semantic similarity with local temporal continuity, thereby partitioning stored historical memory into a set of event-centric EventTrees. For each EventTree, we leverage the **EventTree Progressive Merging** module to perform iterative processing, progressively compressing its internal representations until a predefined compression ratio is achieved. Furthermore, we introduce an **Anchor-Guided Propagation Retrieval (AGPR)** mechanism to recover temporal structural information that may be degraded during compression, thereby enabling the retrieval of more relevant memory nodes. In summary, the main contributions are as follows:

- **General Compression Framework.** Based on the above analysis, we propose **MemForest**, a general agent memory compression framework, which leverages global semantic similarity and local temporal continuity to significantly improve storage and retrieval efficiency.
- **Novel Retrieval Mechanism.** We introduce an **Anchor-Guided Propagation Retrieval** mechanism to retrieve more relevant memory nodes from the temporal neighborhoods of key memory nodes, thereby enabling more accurate memory retrieval.
- **Excellent Empirical Performance.** Under the Mem0 and M3-Agent frameworks, across several benchmarks, when compressing **50%** of the memory, MemForest is able to retain **97.1%** and **99.7%** of the original performance, respectively, while achieving **1.89×** and **2.24×** retrieval speedups.

2 Related Work

2.1 Agent Memory Systems

Agent memory systems [8, 49, 24, 48, 10, 33] aim to provide agents with long-term storage and retrieval capabilities beyond a single context, thereby supporting persistent interaction and complex task execution. According to the type of input data, they can be broadly categorized into unimodal and multimodal approaches. Unimodal methods [31, 35, 58] primarily focus on textual data, such as Mem0 [8], MemoryOS [18], and A-MEM [47], which maintain long-term semantic consistency by storing dialogue histories or structured representations. In contrast, multimodal methods [27, 43, 61] handle more complex inputs, including images, audio, and video. Representative approaches such as MM-MEM [25], M3-Agent [27], and WorldMM [51] integrate visual and linguistic information and further model dynamic scenes such as videos, enabling richer environmental perception and task understanding. However, as memory scales grow, both unimodal and multimodal systems face increasing storage and retrieval costs. This issue is more pronounced in multimodal memory due to greater information redundancy and more complex cross-modal relationships, making efficient memory compression and management a critical challenge.

2.2 Visual Token Compression

Visual token compression [2, 4, 5, 11, 16, 44, 54, 9] aims to reduce the number of tokens while preserving key information as much as possible, thereby lowering computational cost and improving inference efficiency. According to the underlying criteria used for compression, existing methods can be broadly categorized into three types: similarity-based, diversity-based, and attention-based approaches. Similarity-based methods [2, 38] measure the semantic similarity between tokens and merge highly similar ones to reduce redundant representations. Diversity-based methods [44, 54] focus on information deduplication by modeling repetitive relationships among tokens, retaining representative tokens while removing redundant information. Attention-based methods [1, 56, 57] leverage the internal attention mechanism of the model, using attention scores as a measure of token importance to preserve high-weight tokens and discard low-weight ones, enabling adaptive token selection. These methods provide valuable insights for agent memory compression. However, agent memory exhibits unique characteristics such as temporal relationships, and how to effectively leverage these properties for efficient compression remains an open problem.

3 Methodology

3.1 Overview

The overview of our method is illustrated in Figure 2. The MemForest framework consists of two components that achieve efficient compression of historical memory. Meanwhile, the anchor-guided propagation retrieval mechanism enables more accurate retrieval over the compressed memory.

3.2 EventTree Semantic-Temporal Partitioning

Given the historical memory H, which consists of a sequence of memory nodes $m_i = \{c_i, e_i, t_i\} \mid_{i=1}^{N}$, where c_i, e_i, and t_i denote the content, content embedding, and timestamp of the i-th memory node, respectively, our goal is to compress the historical memory H into a compact memory set H', which consists of a sequence of memory nodes $\{m'_j\} \mid_{j=1}^{M}$, where $M < N$.

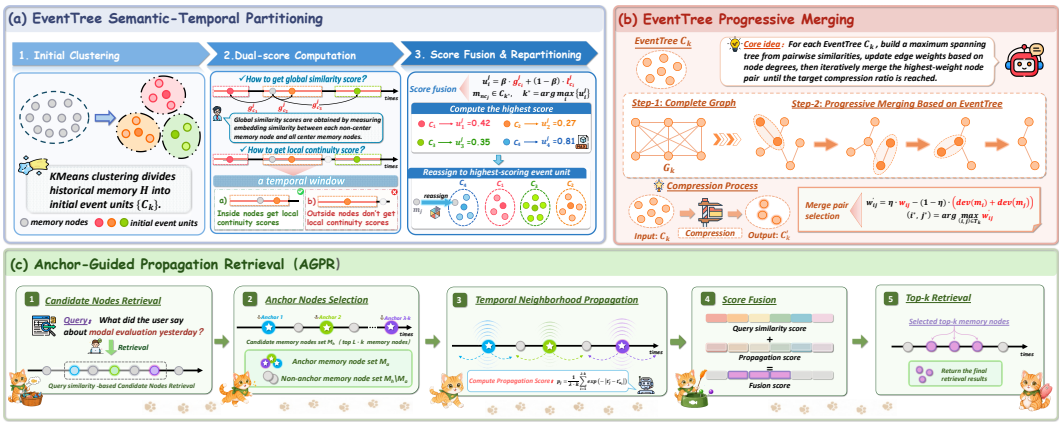


Figure 2: **The overview of our method.** MemForest compresses historical memory through the *EventTree Semantic-Temporal Partitioning* module and the *EventTree Progressive Merging* module, while the *Anchor-Guided Propagation Retrieval* mechanism retrieves more relevant memory nodes within the temporal neighborhoods of key memory nodes, ensuring more accurate retrieval.

First, we perform event-unit partitioning of the historical memory H using the *EventTree Semantic-Temporal Partitioning* module. Specifically, given the initial N memory nodes $\{m_i\}\,|_{i=1}^{N}$, we apply the *KMeans clustering algorithm* [30] to their embedding vectors $\{e_i\}\,|_{i=1}^{N}$ to partition them into $N \cdot \alpha$ initial event units, as shown in the following formula:

$$\{C_k\} = \mathit{KMeans}(\{e_i\}, \alpha), \tag{1}$$

where $\{C_k\}\,|_{k=1}^{N\cdot\alpha}$ denotes the resulting set of event units, and α is a clustering ratio coefficient used to control the number of event units.

Next, for each event unit, we select the memory node closest to the unit center to form the set of center memory nodes H_c, which consists of a sequence of center nodes $\{m_{c_i}\}\,|_{i=1}^{N\cdot\alpha}$. Subsequently, for the j-th non-center memory node $m_{nc_j} \in H \setminus H_c$, we compute its similarity to all center memory nodes, denoted as $\{g_{c_i}^{j}\}\,|_{i=1}^{N\cdot\alpha}$. Each non-center memory node thus obtains $N \cdot \alpha$ similarity values, which serve as its global similarity scores with respect to each center memory node.

Then, the memory nodes in the historical memory H are sorted in chronological order, and the positions of all center memory nodes $\{m_{c_i}\}\,|_{i=1}^{N\cdot\alpha}$ on the timeline are determined. For the i-th center memory node, a local continuity score is assigned to the non-center memory nodes within its neighboring time window, as defined by the following formula:

$$l_{c_i}^{j} = \begin{cases} 1, & \text{if } t_j \in [t_{c_i} - w, t_{c_i} + w] \\ 0, & \text{otherwise} \end{cases}, \tag{2}$$

where w is the time window coefficient that controls the size of the temporal window around the center memory node, and $l_{c_i}^{j}$ represents the local continuity score of the j-th non-center memory node with respect to the i-th center memory node.

Subsequently, the global similarity score $g_{c_i}^{j}$ of the j-th non-center memory node with respect to the i-th center memory node is combined with its local continuity score $l_{c_i}^{j}$ to obtain an overall score:

$$u_i^j = \beta \cdot g_{c_i}^{j} + (1-\beta) \cdot l_{c_i}^{j}, \tag{3}$$

where β is a weighting coefficient that controls the relative contribution of the two scores. Each non-center memory node is then reassigned to the event unit with the highest overall score. Taking the j-th non-center memory node as an example, the assignment can be formulated as:

$$m_{nc_j} \in C_{k^*}, \quad k^* = \arg\max_i \{u_i^j\}, \tag{4}$$

resulting in repartitioned event units. For simplicity, these event units are still denoted as $\{C_k\}\,|_{k=1}^{N\cdot\alpha}$.

3.3 EventTree Progressive Merging

For each event unit, we model it as a maximum spanning tree and perform progressive merging via the *EventTree Progressive Merging* module, thereby representing each event unit as an EventTree.

Specifically, for the k-th EventTree, we denote it as $C_k = \{m_i^{(k)}\}\ |_{i=1}^{n_k}$, where $m_i^{(k)}$ denotes the i-th memory node and n_k is the number of memory nodes in this EventTree. We first construct a fully connected undirected graph $G_k = (V_k, E_k)$, where the node set $V_k = C_k$ and the edge set E_k consists of all node pairs. For any two nodes $m_i^{(k)}$ and $m_j^{(k)}$, the edge weight is defined as the cosine similarity between their embeddings:

$$w_{ij} = \frac{e_i^{(k)} \cdot e_j^{(k)}}{\|e_i^{(k)}\| \cdot \|e_j^{(k)}\|}. \tag{5}$$

We then construct a maximum spanning tree on G_k using *Kruskal algorithm* [19]. Specifically, all edges are first sorted in descending order according to their weights w_{ij}. A union-find structure is used to iteratively add edges with the highest weights while avoiding cycles. This process continues until $n_k - 1$ edges are selected, resulting in the maximum spanning tree T_k.

After obtaining the maximum spanning tree T_k, we update the edge weights by taking into account the degrees of the two nodes connected by each edge, as follows:

$$w'_{ij} = \eta \cdot w_{ij} - (1 - \eta) \cdot \big(\deg(m_i) + \deg(m_j)\big), \tag{6}$$

where $\deg(\cdot)$ represents the degree of a node, and η is a weighting coefficient used to balance the two scores. In the maximum spanning tree, nodes with higher degrees are usually central or hub nodes of the event, carrying more core information, and therefore should not be merged prematurely.

We then select the edge with the highest weight:

$$(i^*, j^*) = \arg \max_{(i,j) \in T_k} w'_{ij}, \tag{7}$$

and treat the corresponding nodes $m_{i^*}^{(k)} = \{c_{i^*}, e_{i^*}, t_{i^*}\}$ and $m_{j^*}^{(k)} = \{c_{j^*}, e_{j^*}, t_{j^*}\}$ as the optimal merge pair at the current step.

Subsequently, we merge the selected node pair by leveraging a large language model to generate fused content c'_l, re-encoding it to obtain the embedding e'_l, and updating the timestamp to t'_l, resulting in a merged memory node $m'_l = \{c'_l,\ e'_l,\ t'_l\}$. The merge process can be formally expressed as:

$$c'_l = \mathcal{F}(c_{i^*}, c_{j^*}), \quad e'_l = \mathcal{E}(c'_l), \quad t'_l = \max(t_{i^*}, t_{j^*}), \tag{8}$$

where $\mathcal{F}(\cdot)$ and $\mathcal{E}(\cdot)$ denote an external large language model and an embedding model, respectively. The newly generated node m'_l then replaces the original nodes $m_{i^*}^{(k)}$ and $m_{j^*}^{(k)}$.

This process is iteratively applied to the updated node set until a predefined compression ratio is reached. Finally, all the compressed EventTrees $\{C'_k\}\ |_{k=1}^{N \cdot \alpha}$ collectively form the compact memory set H', completing the overall compression process.

3.4 Anchor-Guided Propagation Retrieval

To enhance the temporal modeling capability of the compressed memory set H', we propose an *Anchor-Guided Propagation Retrieval* mechanism, which retrieves more relevant memory nodes from the temporal neighborhoods of key memory nodes to supplement nearby information.

Specifically, we first select the top $L \cdot k$ memory nodes with the highest similarity to the query, forming a candidate set M_h, along with their corresponding similarity scores $\{s_i\}\ |_{i=1}^{L \cdot k}$, where L is a scaling coefficient that determines the number of retrieved candidate memory nodes. Then, we further select the top $\lambda \cdot k$ most similar nodes as anchor memory nodes, denoted as $M_a = \{m'_{a_i}\}\ |_{i=1}^{\lambda \cdot k}$.

Next, all memory nodes in M_h are sorted in chronological order, and the positions of anchor memory nodes on the timeline are identified. For the j-th memory node $m'_j \in M_h$, we compute its propagation score with respect to all anchor memory nodes as:

$$p_j = \frac{1}{\lambda \cdot k} \sum_{i=1}^{\lambda \cdot k} \exp\left(-\left|t'_j - t'_{a_i}\right|\right). \tag{9}$$

Finally, we combine the precomputed query similarity and the propagation score to obtain the final fusion score for the j-th memory node:

$$v_j = \gamma \cdot s_j + (1 - \gamma) \cdot p_j, \tag{10}$$

where γ is a weighting coefficient that balances the two scores. We rank all memory nodes in M_h based on v_j and select the top k nodes as the final retrieval results.

4 Theoretical Analysis

We analyze from a retrieval perspective *why similar memory nodes should be preferentially merged.*

Given a query q, consider two memory nodes m_i and m_j with embeddings e_i and e_j, respectively. Their similarities to the query are defined as $s_i = q^\top e_i$ and $s_j = q^\top e_j$, and the similarity between the two nodes is $\rho_{ij} = e_i^\top e_j$. We assume all embeddings are ℓ_2-normalized, i.e., $\|q\| = \|e_i\| = \|e_j\| = 1$. Without loss of generality, we assume $s_i \geq s_j$, indicating that m_i is more relevant to the query q.

We represent the merged memory embedding as a normalized interpolation of the two embeddings:

$$e_l = \frac{\lambda e_i + (1-\lambda) e_j}{\|\lambda e_i + (1-\lambda) e_j\|}, \tag{11}$$

where $\lambda \in (0, 1)$ is a weighting coefficient that controls the interpolation between e_i and e_j.

The similarity between the merged memory node m_l and the query q is given by:

$$s_l = \frac{\lambda s_i + (1-\lambda) s_j}{\|\lambda e_i + (1-\lambda) e_j\|}. \tag{12}$$

By the *Triangle inequality*, we have:

$$s_l \geq \lambda s_i + (1-\lambda) s_j. \tag{13}$$

Furthermore, by the *Cauchy–Schwarz inequality*, it can be shown that:

$$s_j \geq s_i - \sqrt{2 - 2\rho_{ij}}, \tag{14}$$

which leads to:

$$s_l \geq s_i - (1-\lambda)\sqrt{2 - 2\rho_{ij}}. \tag{15}$$

The derivation of Eqs. 13-15 is provided in Appendix B.1. From Eq. 15, we observe that as ρ_{ij} increases, the lower bound monotonically increases, indicating that merging highly similar memory nodes preserves higher query similarity. In contrast, when ρ_{ij} is small, the lower bound decreases, implying larger semantic deviation after merging.

During top k retrieval, such deviation may prevent important memory nodes from being correctly retrieved. Therefore, it is preferable to prioritize merging highly similar memory nodes.

Furthermore, for the theoretical analysis of MemForest, please refer to Appendix B.2.

5 Experiments

5.1 Experimental Setup

Baselines. For unimodal and multimodal memory systems, we select Mem0 [8] and M3-Agent [27] as representative methods, respectively. For the baselines, we categorize them into two groups: pruning-based methods and merging-based methods. Specifically, pruning-based methods include Random Pruning, KMeans [30], DART [44], and StreamMeCo [41]; merging-based methods include Random Merging and ToMe [2]. More details on these methods are provided in Appendix A.1.

Datasets. For Mem0 [8], we evaluate on three representative benchmarks in the agent memory domain, including LoCoMo [28], LongMemEval [45], and PersonaMem [17]. For M3-Agent [27], we adopt two representative benchmarks from the streaming video domain, including M3-Bench-robot and M3-Bench-web [27]. More details on these benchmarks are provided in Appendix A.2.

Implementation Details. All experiments are conducted on two NVIDIA A100 (80GB) GPUs. The key hyperparameters are set as follows: $\alpha = 0.05$, $\beta = 0.8$, $w = 5$, $\eta = 0.99$, $\gamma = 0.9$, $L = 4$, and $\lambda = 0.2$. We use GPT-4o-mini for memory node merging. In addition, all other settings follow the original papers [8, 27]. All experimental results are averaged over three runs. More details are provided in Appendix A.3.

Table 1: **Performance comparison of different baselines on Mem0 under varying compression ratios.** Blue denotes pruning methods, while Red denotes merging methods. **%** represents the performance retention rate. For each ratio, the top two results are highlighted in **bold black font**.

Dataset	LoCoMo					LongMemEval							Persona-Mem	Avg.	%
Method	SH	MH	TR	OD	**All**	SU	SA	SP	MS	KU	TR	**All**			
Mem0	66.0	56.0	53.9	42.7	60.2	90.0	21.4	23.3	54.9	79.5	44.4	55.2	67.9	61.1	100.0%
+ AGPR	66.2	56.4	53.9	40.6	60.3	90.0	21.4	23.3	54.1	78.2	43.6	54.6	68.6	61.2	100.2%
Historical memory compression ratio (↓ 30%)															
+ Random Pruning	55.3	49.7	43.3	39.6	50.8	82.9	17.9	26.7	51.9	73.1	39.1	50.8	64.5	55.4	90.7%
+ KMeans	61.1	54.3	52.7	40.6	56.8	84.3	17.9	20.0	51.9	75.6	41.4	52.0	65.7	58.2	95.3%
+ DART	64.1	53.9	48.0	40.6	57.4	82.9	19.0	20.0	48.9	74.4	34.6	48.8	**66.6**	57.6	94.3%
+ Random Merging	60.9	52.1	51.1	43.8	56.2	91.4	21.4	20.2	46.6	74.4	41.4	50.8	65.2	57.4	93.9%
+ ToMe	63.1	51.1	50.8	44.8	57.2	85.7	21.4	23.3	48.1	71.8	42.9	51.2	66.4	58.3	95.4%
+ MemForest	63.7	54.6	52.0	45.8	**58.5**	90.0	19.6	20.0	49.8	76.9	49.6	**54.2**	66.4	**59.7**	**97.7%**
+ MemForest + AGPR	64.6	54.6	52.3	44.8	**59.0**	90.0	19.6	20.0	52.6	78.8	43.6	**53.8**	**67.6**	**60.1**	**98.4%**
Historical memory compression ratio (↓ 50%)															
+ Random Pruning	44.4	45.4	37.1	44.8	43.1	75.7	17.9	26.7	32.3	65.4	39.8	43.6	64.3	50.3	82.3%
+ KMeans	51.6	43.6	39.6	37.5	46.8	77.1	19.6	23.3	39.1	74.4	39.8	47.0	65.2	53.0	86.7%
+ DART	57.4	49.6	41.7	43.8	51.9	67.1	12.5	23.3	42.9	60.3	31.6	41.4	64.3	52.5	85.9%
+ Random Merging	56.0	48.2	46.4	37.5	51.4	85.7	23.2	20.0	42.9	70.5	39.1	48.6	64.5	54.8	89.7%
+ ToMe	60.2	50.4	47.4	43.8	54.7	85.7	21.4	23.3	48.1	71.8	41.4	50.8	65.9	57.1	93.5%
+ MemForest	62.1	56.7	54.8	43.8	**58.4**	90.0	19.6	20.0	48.1	74.4	42.9	**51.8**	**67.7**	**59.3**	**97.1%**
+ MemForest + AGPR	63.7	57.1	52.3	45.8	**59.0**	90.0	21.4	23.3	47.4	76.9	45.1	**53.0**	**68.1**	**60.0**	**98.3%**
Historical memory compression ratio (↓ 70%)															
+ Random Pruning	35.0	36.5	25.9	31.3	33.1	47.1	7.1	30.0	16.5	42.3	31.6	28.6	63.7	41.8	68.4%
+ KMeans	37.9	42.6	27.4	39.6	36.7	61.4	8.9	30.0	21.8	59.0	31.6	34.8	63.8	45.1	73.8%
+ DART	46.4	44.3	36.1	41.7	43.6	44.3	10.7	16.7	21.0	43.6	27.8	30.6	61.8	45.3	74.1%
+ Random Merging	54.0	47.2	47.0	43.8	50.7	80.0	21.4	30.0	41.4	67.9	34.6	46.2	64.0	53.6	87.7%
+ ToMe	56.6	51.8	47.7	38.5	52.7	84.3	21.4	20.0	45.1	70.5	42.1	49.6	**65.7**	56.0	91.7%
+ MemForest	58.1	56.4	45.2	45.8	**54.4**	84.3	25.0	23.3	52.6	74.4	39.8	**52.2**	64.4	**57.0**	**93.3%**
+ MemForest + AGPR	57.7	55.0	45.6	45.8	**53.9**	85.7	26.8	26.7	51.1	74.4	39.8	**52.4**	**65.5**	**57.3**	**93.8%**

5.2 Main Results

Performance under the Mem0 Framework. Table 1 reports the performance of various baselines under the Mem0 framework. The experimental results can be summarized in three points: **(i) Outstanding performance.** Without incorporating the AGPR mechanism, MemForest is able to retain **97.1%** of the original performance across three representative benchmarks even when **50%** of historical memory is compressed, significantly outperforming various pruning-based and merging-based baselines. **(ii) Historical information preservation.** At a low compression ratio (30%), pruning-based and merging-based methods perform comparably. However, when the compression ratio exceeds **50%**, pruning-based methods suffer a noticeable performance drop, whereas merging-based methods still maintain relatively strong performance and preserve more historical information. **(iii) Dataset-specific differences.** On the PersonaMem benchmark, all methods exhibit relatively minor performance degradation under different compression ratios. We attribute this to the multiple-choice nature of the questions, which requires only limited memory for decision-making.

Performance under the M3-Agent Framework. Table 2 presents the performance of various baselines under the M3-Agent framework. The experimental results can be summarized as follows: **(i) Outstanding performance.** Without incorporating the AGPR mechanism, MemForest is able to retain **99.7%** of the original performance across two representative benchmarks even when **50%** of historical memory is compressed, significantly outperforming various pruning-based and merging-based baselines. **(ii) Historical information preservation.** When the compression ratio exceeds **50%**, even the memory pruning framework StreamMeCo, which is specifically designed for M3-Agent, performs worse than random compression, indicating that merging-based methods better preserve historical information. **(iii) Higher redundancy.** Compared with Mem0, the performance degradation of M3-Agent is considerably smaller, indicating that multimodal memory with dense visual inputs has higher redundancy than unimodal text memory, offering greater potential for compression.

Table 2: **Performance comparison of different baselines on M3-Agent under varying compression ratios.** Blue denotes pruning methods, while Red denotes merging methods. **%** represents the performance retention rate. For each ratio, the top two results are highlighted in **bold black font**.

Dataset	M3-Bench-robot						M3-Bench-web						Avg.	%
Method	ME	MH	CM	PU	GK	**All**	ME	MH	CM	PU	GK	**All**		
M3-Agent	30.9	29.4	29.6	41.1	22.6	30.3	44.9	25.6	44.8	58.6	53.7	47.9	39.1	100.0%
+ AGPR	37.2	38.8	34.2	47.3	29.7	37.2	49.2	35.9	49.8	64.5	61.2	54.8	46.0	117.6%
Historical memory compression ratio (↓ 30%)														
+ Random Pruning	31.4	27.1	29.8	42.5	18.3	29.1	40.8	23.9	41.7	56.0	49.1	44.9	37.0	94.6%
+ KMeans	31.6	32.9	29.6	42.2	19.3	30.2	41.0	24.1	45.0	56.1	49.8	45.8	38.0	97.2%
+ DART	31.8	29.4	32.6	43.8	19.3	29.6	40.9	24.9	43.4	58.2	49.6	46.1	37.8	96.7%
+ StreamMeCo	32.9	30.6	30.9	42.5	19.6	30.7	41.3	26.0	44.3	58.3	50.5	47.0	38.9	99.5%
+ Random Merging	31.7	35.3	30.3	40.9	19.9	30.0	40.4	23.2	42.9	55.9	50.1	45.1	37.6	96.2%
+ ToMe	30.8	25.9	27.7	40.5	21.7	30.4	42.4	27.6	41.7	57.5	52.0	47.1	38.8	99.2%
+ MemForest	32.3	27.1	33.6	43.8	22.6	**32.0**	43.1	26.3	44.6	57.9	52.2	**47.2**	**39.6**	**101.3%**
+ MemForest + AGPR	39.4	37.6	36.8	50.5	29.4	**39.0**	50.9	39.2	52.4	64.0	63.0	**56.0**	**47.5**	**121.5%**
Historical memory compression ratio (↓ 50%)														
+ Random Pruning	28.4	25.9	27.3	38.7	19.6	28.1	35.6	23.0	34.9	52.3	44.2	41.2	34.7	88.7%
+ KMeans	29.1	30.6	28.4	41.4	20.8	29.2	41.1	21.7	42.5	55.0	47.4	43.9	36.6	93.6%
+ DART	28.9	25.9	29.4	39.1	22.0	29.1	39.9	23.6	39.4	52.9	47.2	43.1	36.1	92.3%
+ StreamMeCo	32.3	28.2	29.8	41.4	21.1	30.6	39.7	25.2	38.9	58.4	47.3	44.7	37.7	96.2%
+ Random Merging	33.0	32.9	31.9	44.3	19.6	30.8	41.8	21.0	40.1	58.0	46.3	44.6	37.7	96.4%
+ ToMe	30.5	32.9	29.6	42.3	19.9	30.9	42.7	24.7	41.7	54.3	53.1	46.1	38.5	98.5%
+ MemForest	31.6	31.8	31.1	45.1	21.4	**31.6**	42.6	23.9	43.4	57.1	51.5	**46.5**	**39.0**	**99.7%**
+ MemForest + AGPR	40.5	38.8	36.1	48.2	31.8	**38.3**	52.8	34.4	50.0	65.0	63.8	**56.1**	**47.2**	**120.7%**
Historical memory compression ratio (↓ 70%)														
+ Random Pruning	27.3	25.9	26.7	39.2	20.8	27.7	35.0	19.9	34.9	48.7	39.5	37.3	32.5	83.1%
+ KMeans	28.7	28.2	27.3	42.7	19.3	28.9	35.5	20.1	34.9	39.7	42.5	39.1	34.0	87.0%
+ DART	27.8	25.9	25.4	40.0	16.2	27.2	35.4	19.5	36.6	48.2	45.0	39.2	33.2	84.9%
+ StreamMeCo	29.1	30.6	26.9	40.3	19.9	28.4	37.0	21.7	36.1	53.0	46.3	41.9	35.1	89.8%
+ Random Merging	30.6	31.8	28.8	45.1	19.0	30.4	41.1	21.9	39.6	58.6	41.5	43.2	36.8	94.1%
+ ToMe	31.7	27.1	28.6	43.1	20.2	30.6	42.9	23.6	43.6	56.5	49.8	45.7	38.2	97.7%
+ MemForest	31.2	32.9	32.4	43.6	17.4	**30.7**	44.8	23.9	38.9	57.3	50.4	**46.3**	**38.5**	**98.5%**
+ MemForest + AGPR	38.5	38.8	35.7	48.2	29.7	**37.5**	52.8	34.1	50.0	64.1	64.3	**55.8**	**46.7**	**119.4%**

Performance of the AGPR Mechanism. Tables 1–2 illustrate the performance of the AGPR mechanism across the two memory frameworks. The experimental results can be summarized in three points: **(i) Outstanding performance.** The AGPR mechanism consistently improves performance across different compression ratios in both memory frameworks. **(ii) Framework-dependent performance gains.** Under the Mem0 framework, the performance improvement brought by AGPR is relatively limited and noticeably smaller than that in M3-Agent. This difference mainly stems from differences in the original retrieval mechanism and the nature of the input data. The original retrieval mechanism of M3-Agent is relatively weak; in addition, Mem0 uses dense textual input with low redundancy, whereas M3-Agent contains multimodal input with substantial redundancy, allowing AGPR to retrieve key memory nodes more accurately. **(iii) Information loss compensation.** Performance gains are more pronounced after compressing historical memory than without compression. For example, when **50%** of historical memory is compressed, performance improvements in Mem0 and M3-Agent are **1.2%** and **21.0%**, respectively, compared to **0.2%** and **17.6%** without compression. This demonstrates that AGPR can effectively compensate for information loss by enriching the contextual information around key memory nodes.

5.3 Ablation Study

5.3.1 MemForest Ablation Study.

EventTree Semantic-Temporal Partitioning. Table 3 suggests that, when compressing **50%** of historical memory, our method significantly outperforms two variants across three benchmarks: one

Table 3: **Impact of different EventTree partitioning strategies at 50% compression rate.**

Method	LoCoMo	LongMemEval	M3-Bench-web	Avg.	%
Baseline	60.2	55.2	47.9	54.4	100.0%
w/o $g^j_{c_i}$	54.5	49.2	45.6	49.8	91.5%
w/o $l^j_{c_i}$	56.4	50.2	46.2	50.9	93.6%
Ours	**58.4**	**51.8**	**46.5**	**52.2**	**96.0%**

Table 4: **Impact of different memory node merging strategies at 50% compression rate.**

Method	LoCoMo	LongMemEval	M3-Bench-web	Avg.	%
Baseline	60.2	55.2	47.9	54.4	100.0%
Random	54.0	50.0	45.7	49.9	91.7%
Minimum	51.2	48.8	45.2	48.4	89.0%
Ours	**58.4**	**51.8**	**46.5**	**52.2**	**96.0%**

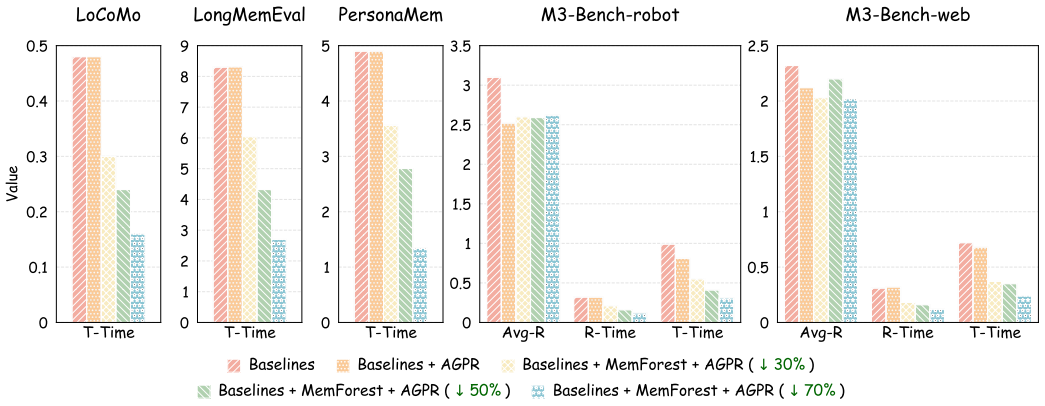


Figure 3: **Time Efficiency Analysis on Mem0 and M3-Agent**. *Avg-R(×)*: average number of retrievals per query; *R-Time(s)*: time cost per retrieval; *T-Time(s)*: total retrieval time per query.

removes the local continuity score $l^j_{c_i}$, and the other removes the global similarity score $g^j_{c_i}$ and replaces it with a sliding window for EventTree partitioning. Detailed implementations of these variants are provided in Appendix A.3. This indicates that combining global semantic similarity with local temporal continuity enables more accurate EventTree partitioning.

EventTree Progressive Merging. Table 4 suggests that, when compressing **50%** of the historical memory, replacing the maximum spanning tree with either random merging or minimum spanning tree merging results in significantly worse performance than our method, thereby validating the theoretical analysis in Section 4.

5.3.2 Time Efficiency Analysis.

Figure 3 suggests our proposed method achieves high time efficiency, and detailed results can be found in Appendix D.7, from which we draw three key observations: **(i) Minimal time overhead.** Since the AGPR mechanism avoids complex embedding similarity computations, it introduces negligible additional time overhead. **(ii) Significant speedup.** Under the Mem0 framework, when compressing **50%** of historical memory across three benchmarks, our method achieves speedups of **2.00**×, **1.92**×, and **1.76**×, respectively, with an average speedup of **1.89**×. **(iii) More accurate retrieval.** M3-Agent typically requires multiple rounds of memory retrieval for a single query to obtain sufficient information, whereas our method significantly reduces the number of retrieval iterations, indicating more accurate retrieval of critical memory nodes. As a result, our method achieves speedups of **2.41**× and **2.06**× on two benchmarks, respectively, with an average speedup of **2.24**×.

For more ablation studies, additional evaluation metrics, parameter sensitivity analyses, merge cost analysis, analysis of merging effectiveness across different models, case studies, discussions on future work, etc., please refer to Appendices D-G.

6 Conclusion

We propose MemForest, a general agent memory compression framework, and introduce an anchor-guided propagation retrieval mechanism to retrieve more relevant neighborhood information. Extensive experiments demonstrate the effectiveness of both components. Under the Mem0 and M3-Agent frameworks, when compressing 50% of historical memory, MemForest retains 97.1% and 99.7% of the original performance, while achieving 1.89× and 2.24× retrieval speedups, respectively.

Acknowledgments

This paper was partially sponsored by WUYING - Alibaba Cloud.

References

[1] Kazi Hasan Ibn Arif, JinYi Yoon, Dimitrios S Nikolopoulos, Hans Vandierendonck, Deepu John, and Bo Ji. Hired: Attention-guided token dropping for efficient inference of high-resolution vision-language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 1773–1781, 2025.

[2] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*, 2022.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[4] Joya Chen, Zhaoyang Lv, Shiwei Wu, Kevin Qinghong Lin, Chenan Song, Difei Gao, Jia-Wei Liu, Ziteng Gao, Dongxing Mao, and Mike Zheng Shou. Videollm-online: Online video large language model for streaming video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18407–18418, 2024.

[5] Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision*, pages 19–35. Springer, 2024.

[6] Shuguang Chen and Guang Lin. Llm reasoning engine: Specialized training for enhanced mathematical reasoning. In *Proceedings of the 4th International Workshop on Knowledge-Augmented Methods for Natural Language Processing*, pages 118–128, 2025.

[7] Xueyi Chen, Keda Tao, Kele Shao, and Huan Wang. Streamingtom: Streaming token compression for efficient video understanding. *arXiv preprint arXiv:2510.18269*, 2025.

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

[9] Dasen Dai, Shuoqi Li, Ronghao Chen, Huacan Wang, Biao Wu, and Qizhen Lan. Uipress: Bringing optical token compression to ui-to-code generation. *arXiv preprint arXiv:2604.09442*, 2026.

[10] Xingjian Diao, Wenbo Li, Yashas Malur Saidutta, Avinash Amballa, Lazar Valkov, and Srinivas Chappidi. Doc-to-atom: Learning to compile and compose memory atoms. *arXiv preprint arXiv:2606.12400*, 2026.

[11] Vaggelis Dorovatas, Soroush Seifi, Gunshi Gupta, and Rahaf Aljundi. Recurrent attention-based token selection for efficient streaming video-llms. *arXiv preprint arXiv:2510.17364*, 2025.

[12] Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, et al. Lightmem: Lightweight and efficient memory-augmented generation. *arXiv preprint arXiv:2510.18866*, 2025.

[13] Zhen Fang, Yu Zeng, Wenxuan Huang, Yiming Zhao, Shiting Huang, Tianfei Ren, Qi Lu, Qingnan Ren, Qisheng Su, Lionel Z. Wang, Qingyu Yin, Shuang Chen, Zehui Chen, Lin Chen, Zhenfei Yin, Yao Hu, Shaohui Lin, Wanli Ouyang, Shaosheng Cao, and Feng Zhao. Video-deepresearch: Towards the next-generation multimodal deepresearch agent, 2026.

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

[15] Zhanghao Hu, Qinglin Zhu, Di Liang, Hanqi Yan, Yulan He, and Lin Gui. Beyond rag for agent memory: Retrieval by decoupling and aggregation, 2026.

[16] Zhenpeng Huang, Xinhao Li, Jiaqi Li, Jing Wang, Xiangyu Zeng, Cheng Liang, Tao Wu, Xi Chen, Liang Li, and Limin Wang. Online video understanding: Ovbench and videochat-online. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 3328–3338, 2025.

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

[18] Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory os of ai agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 25972–25981, 2025.

[19] Joseph B Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.

[20] Chao Lei, Yanchuan Chang, Nir Lipovetzky, and Krista A Ehinger. Planning-driven programming: A large language model programming workflow. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12647–12684, 2025.

[21] Xiaoyuan Li, Yubo Ma, Chengpeng Li, Fengbin Zhu, Yiyao Yu, Keqin Bao, Wenjie Wang, Fuli Feng, and Dayiheng Liu. Unified data selection for LLM reasoning. *CoRR*, abs/2605.22389, 2026.

[22] Xiaoyuan Li, Yuzhe Wang, Moxin Li, Keqin Bao, Rui Men, Yichang Zhang, Dayiheng Liu, Wenjie Wang, and Fuli Feng. Sage: Scalable automated robustness augmentation for llm knowledge evaluation. *arXiv preprint arXiv:2605.12022*, 2026.

[23] Yue Li, Mikael Johansson, and Andrey R Nikolaev. Hierarchical event segmentation of episodic memory in virtual reality. *npj Science of Learning*, 10(1):25, 2025.

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

[25] Niu Lian, Yuting Wang, Hanshu Yao, Jinpeng Wang, Bin Chen, Yaowei Wang, Min Zhang, and Shu-Tao Xia. From verbatim to gist: Distilling pyramidal multimodal memory via semantic information bottleneck for long-horizon video agents. *arXiv preprint arXiv:2603.01455*, 2026.

[26] Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. Simplemem: Efficient lifelong memory for llm agents. *arXiv preprint arXiv:2601.02553*, 2026.

[27] Lin Long, Yichen He, Wentao Ye, Yiyuan Pan, Yuan Lin, Hang Li, Junbo Zhao, and Wei Li. Seeing, listening, remembering, and reasoning: A multimodal agent with long-term memory. *arXiv preprint arXiv:2508.09736*, 2025.

[28] 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.

[29] David Marr. *Vision: A computational investigation into the human representation and processing of visual information*. MIT press, 2010.

[30] James B McQueen. Some methods of classification and analysis of multivariate observations. In *Proc. of 5th berkeley symposium on math. stat. and prob.*, pages 281–297, 1967.

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

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

[33] Peijun Qing, Xingjian Diao, Chiyu Ma, Saeed Hassanpour, and Soroush Vosoughi. Tailoring memory granularity for multi-hop reasoning over long contexts. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 3648–3666, 2026.

[34] Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025.

[35] 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.

[36] Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Eric Michael Smith, Y-Lan Boureau, et al. Recipes for building an open-domain chatbot. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 300–325, 2021.

[37] Samarth Sarin, Lovepreet Singh, Bhaskarjit Sarmah, and Dhagash Mehta. Memoria: A scalable agentic memory framework for personalized conversational ai. In *2025 5th International Conference on AI-ML-Systems (AIMLSystems)*, pages 32–39. IEEE, 2025.

[38] Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. Llava-prumerge: Adaptive token reduction for efficient large multimodal models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22857–22867, 2025.

[39] Maverick E Smith and Jeffrey M Zacks. Event segmentation interventions improve memory for naturalistic events. *Current Directions in Psychological Science*, 35(1):33–40, 2026.

[40] Jinwei Su, Qizhen Lan, Yinghui Xia, Lifan Sun, Weiyou Tian, Tianyu Shi, and Lewei He. Difficulty-aware agentic orchestration for query-specific multi-agent workflows. In *Proceedings of the ACM Web Conference 2026*, pages 2060–2070, 2026.

[41] Junxi Wang, Te Sun, Jiayi Zhu, Junxian Li, Haowen Xu, Zichen Wen, Xuming Hu, Zhiyu Li, and Linfeng Zhang. Streammeco: Long-term agent memory compression for efficient streaming video understanding. *arXiv preprint arXiv:2604.09000*, 2026.

[42] Piaohong Wang, Motong Tian, Jiaxian Li, Yuan Liang, Yuqing Wang, Qianben Chen, Tiannan Wang, Zhicong Lu, Jiawei Ma, Yuchen Eleanor Jiang, et al. O-mem: Omni memory system for personalized, long horizon, self-evolving agents. *arXiv preprint arXiv:2511.13593*, 2025.

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

[44] Zichen Wen, Yifeng Gao, Shaobo Wang, Junyuan Zhang, Qintong Zhang, Weijia Li, Conghui He, and Linfeng Zhang. Stop looking for "important tokens" in multimodal language models: Duplication matters more. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 9972–9991, 2025.

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

[46] Tingyu Wu, Zhisheng Chen, Ziyan Weng, Shuhe Wang, Shuo Zhang, Sen Hu, Silin Wu, Qizhen Lan, Huacan Wang, and Ronghao Chen. Knowme-bench: Benchmarking person understanding for lifelong digital companions. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 30214–30238, 2026.

[47] 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.

[48] BY Yan, Chaofan Li, Hongjin Qian, Shuqi Lu, and Zheng Liu. General agentic memory via deep research. *arXiv preprint arXiv:2511.18423*, 2025.

[49] Tianyu Yang, Sudipta Paul, Vijay Srinivasan, Vivek Kulkarni, and Srinivas Chappidi. Trustmem: Learning trustworthy memory consolidation for llm agents with long-term memory. *arXiv preprint arXiv:2606.25161*, 2026.

[50] Linli Yao, Yicheng Li, Yuancheng Wei, Lei Li, Shuhuai Ren, Yuanxin Liu, Kun Ouyang, Lean Wang, Shicheng Li, Sida Li, et al. Timechat-online: 80% visual tokens are naturally redundant in streaming videos. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 10807–10816, 2025.

[51] Woongyeong Yeo, Kangsan Kim, Jaehong Yoon, and Sung Ju Hwang. Worldmm: Dynamic multimodal memory agent for long video reasoning. *arXiv preprint arXiv:2512.02425*, 2025.

[52] Xiangyu Zeng, Kefan Qiu, Qingyu Zhang, Xinhao Li, Jing Wang, Jiaxin Li, Ziang Yan, Kun Tian, Meng Tian, Xinhai Zhao, et al. Streamforest: Efficient online video understanding with persistent event memory. *arXiv preprint arXiv:2509.24871*, 2025.

[53] Yu Zeng, Wenxuan Huang, Zhen Fang, Shuang Chen, Yufan Shen, Yishuo Cai, Xiaoman Wang, Zhenfei Yin, Lin Chen, Zehui Chen, et al. Vision-deepresearch benchmark: Rethinking visual and textual search for multimodal large language models. *arXiv preprint arXiv:2602.02185*, 2026.

[54] Evelyn Zhang, Fufu Yu, Aoqi Wu, Zichen Wen, Ke Yan, Shouhong Ding, Biqing Qi, and Linfeng Zhang. D^2pruner: Debiased importance and structural diversity for mllm token pruning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 12412–12420, 2026.

[55] Qizhe Zhang, Aosong Cheng, Ming Lu, Renrui Zhang, Zhiyong Zhuo, Jiajun Cao, Shaobo Guo, Qi She, and Shanghang Zhang. Beyond text-visual attention: Exploiting visual cues for effective token pruning in vlms. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 20857–20867, 2025.

[56] Qizhe Zhang, Aosong Cheng, Ming Lu, Zhiyong Zhuo, Minqi Wang, Jiajun Cao, Shaobo Guo, Qi She, and Shanghang Zhang. [cls] attention is all you need for training-free visual token pruning: Make vlm inference faster. *arXiv e-prints*, pages arXiv–2412, 2024.

[57] Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, et al. Sparsevlm: Visual token sparsification for efficient vision-language model inference. *arXiv preprint arXiv:2410.04417*, 2024.

[58] 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.

[59] Chunzheng Zhu, Yangfang Lin, Shen Chen, Yijun Wang, and Jianxin Lin. Medeyes: Learning dynamic visual focus for medical progressive diagnosis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 13916–13924, 2026.

[60] Chunzheng Zhu, Lei Tian, Bohan Tan, Ziqi Zhou, Yuxuan Sun, Yijun Wang, Chengchao Lv, Yilin Wen, Yijun He, Jinghao Lin, et al. The path to self-evolving clinical systems: Scaling medical agents from assistance to autonomy. *arXiv preprint arXiv:2607.11175*, 2026.

[61] Wazeer Deen Zulfikar, Samantha Chan, and Pattie Maes. Memoro: Using large language models to realize a concise interface for real-time memory augmentation. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, pages 1–18, 2024.

Appendix

A Other Experimental Details **14**

A.1 Baselines 14

A.2 Datasets 15

A.3 Additional Implementation Details 16

B Supplementary Theoretical Analysis **16**

B.1 Theoretical Analysis of Why Similar Memory Nodes Should Be Merged 16

B.2 Theoretical Analysis of MemForest 17

C Memory Node Merging Prompt **17**

D Additional Experiments **18**

D.1 Ablation Study under Different Ratios 18

D.2 Other Ablation Study 18

D.3 Other Evaluation Metrics 20

D.4 Parameter Sensitivity Analysis 20

D.5 Memory Node Merging Cost Analysis 21

D.6 Merging Performance across Different Models 21

D.7 Detailed time efficiency results 22

E Case Study **22**

F Limitations **24**

G Future Work **24**

A Other Experimental Details

A.1 Baselines

Mem0. Mem0 [8] is a long-term memory framework for large language models that addresses the limitation of fixed context windows by dynamically extracting, consolidating, and retrieving key information from conversations. It consists of two stages, memory extraction and memory update, and can automatically decide whether to add, update or delete information to maintain consistency and effectiveness of memory. Overall, it significantly reduces computational overhead while preserving reasoning ability, enabling AI agents to achieve stronger long-term interaction capability.

M3-Agent. M3-Agent [27] is a multimodal agent framework that continuously perceives video and audio inputs and builds long-term memory to accumulate environmental knowledge. Its memory is organized in an entity-centric manner, integrating visual, auditory, and textual information to achieve more consistent understanding and representation. During task execution, it performs multi-turn reasoning and memory retrieval collaboratively to handle complex tasks more effectively. Overall, it enables multimodal agents to achieve more human-like perception, memory, and reasoning capabilities.

Random Pruning. Regarding Random Pruning, we randomly select memory nodes for pruning according to the specified compression ratio.

KMeans. Regarding KMeans [30], we cluster the memory nodes using the KMeans algorithm and according to the desired compression ratio, partition them into a set of clusters. From each cluster, the memory node closest to the centroid is retained, while the remaining nodes are pruned.

DART. DART [44] is a diversity-based token pruning approach that reduces computational overhead by identifying and removing highly redundant visual tokens. Its core idea is to select a small set of pivot tokens and preferentially retain tokens with low similarity to them, thereby preserving key information during compression. The method requires no additional training and is compatible with efficient attention mechanisms, achieving strong performance even under aggressive token reduction. We extend it to the agent memory compression setting: following the original setup, we randomly select 2% of tokens as pivot tokens, compute the overall similarity between remaining tokens and these pivots, and retain those with the lowest similarity.

StreamMeCo. StreamMeCo [41] is a memory compression method specifically designed for M3-Agent [27], addressing the storage and retrieval bottlenecks caused by the growth of memory graphs in streaming video scenarios by efficiently compressing memory nodes in a structure-aware manner. Its core adopts a dual-branch strategy, performing representative sampling for isolated nodes and pruning connected nodes by jointly considering entity importance and semantic similarity, thereby preserving key information during compression.

Random Merging. Regarding Random Merging, we partition the memory nodes into two sets according to the compression ratio, where the number of nodes in the second set corresponds to the number of nodes to be retained. We then randomly merge nodes from the second set with those in the first set until the target compression ratio is achieved.

ToMe. ToMe [2] is a similarity-based token merging approach that reduces computational cost by progressively merging similar tokens within Transformer layers. Its core idea is to compute pairwise similarity between tokens and perform matching, merging similar tokens into a single representation, thereby preserving key information while reducing the number of tokens. The method can be applied directly without additional training and achieves a favorable trade-off between efficiency and performance across multimodal tasks. We extend it to the agent memory compression setting: following the original design, memory nodes are partitioned into two equally sized sets in an alternating manner based on their insertion order, and for each node in the first set, the most similar node in the second set is identified and merged. Each partitioning step can compress up to 50% of the historical memory; if the target compression ratio is not reached, the process is iteratively repeated.

A.2 Datasets

LoCoMo. The LoCoMo dataset [28] consists of 10 very long conversations, each with about 27 sessions, 600 turns, and 16K tokens, designed to simulate long-term interactions. It includes 1,986 question answering instances across five categories, including single hop, multi hop, temporal reasoning, commonsense reasoning, and adversarial questions, to evaluate models' long term memory and reasoning abilities.

LongMemEval. The LongMemEval dataset [45] is a benchmark for evaluating long-term interactive memory, consisting of 500 user–assistant multi-session dialogue histories corresponding to 500 questions. Each dialogue is composed of multiple sessions, and the questions are categorized into six types, including single-session-user, single-session-assistant, single-session-preference, multi-session, knowledge-update, and temporal-reasoning, to assess models' long-term memory and reasoning capabilities.

PersonaMem. The PersonaMem dataset [17] provides multiple context-scale versions, including 32k, 128k, and 1M tokens. In this work, we adopt the PersonaMem-32k version, which contains 222 user–model multi-session dialogue histories and 589 questions. The question types mainly cover user fact recall, new idea generation, latest preference identification, preference evolution tracking, reasoning about preference changes, preference-aligned recommendation, and cross-scenario generalization, to evaluate models' long-term personalized memory and reasoning capabilities.

M3-Bench-robot. The M3-Bench-robot dataset [27] is an online video dataset consisting of 100 first-person robot videos across seven daily environments such as living rooms, kitchens, and offices, with a total of 1,276 questions. Each video involves interactions between the robot and multiple humans, requiring long-term memory construction and reasoning. The dataset includes five task types: multi-evidence reasoning, multi-hop reasoning, cross-modal reasoning, person understanding, and general knowledge extraction, to evaluate memory and reasoning in dynamic interactions.

M3-Bench-web. The M3-Bench-web dataset [27] is an offline video dataset with 920 YouTube videos across 46 categories such as documentaries, travel, and sports, containing 3,214 questions. It includes five task types: multi-evidence reasoning, multi-hop reasoning, cross-modal reasoning, person understanding, and general knowledge extraction, to evaluate models' cross-modal understanding and long-term memory reasoning.

A.3 Additional Implementation Details

During the experiments, all settings for both the Mem0 and M3-Agent frameworks strictly follow their original papers [8, 27]. For the Mem0 framework, we use GPT-4o-mini to evaluate the overall answer quality, while for the M3-Agent framework, we employ GPT-4o for evaluation. All memory node merging processes are conducted using GPT-4o-mini. In terms of embedding models, Mem0 uses text-embedding-3-small, whereas M3-Agent adopts text-embedding-3-large.

In addition, in Section 5.3, we use a sliding window approach: all memory nodes are sorted chronologically with a window size of 5. If the average embedding similarity between two adjacent windows exceeds 0.5, they are assigned to the same EventTree; otherwise, a new EventTree is initiated starting from the latter window.

B Supplementary Theoretical Analysis

B.1 Theoretical Analysis of Why Similar Memory Nodes Should Be Merged

In this section, we provide a detailed derivation of Eqs. 13-15 in Section 4 for completeness. To facilitate the following analysis, we first introduce two basic lemmas.

Lemma 1 (Triangle Inequality). For any vectors x and y, we have

$$\|x + y\| \leq \|x\| + \|y\|. \tag{16}$$

Lemma 2 (Cauchy–Schwarz Inequality). For any vectors x and y, we have

$$x^\top y \leq \|x\| \cdot \|y\|. \tag{17}$$

Proof:

By definition, the similarity between the merged memory node m_l and the query q is given by:

$$s_l = \frac{\lambda s_i + (1-\lambda) s_j}{\|\lambda e_i + (1-\lambda) e_j\|}. \tag{18}$$

To bound the denominator, we apply Lemma 1, which gives:

$$\|\lambda e_i + (1-\lambda) e_j\| \leq \lambda \|e_i\| + (1-\lambda)\|e_j\| = 1. \tag{19}$$

Combining the above results, we obtain the following lower bound on s_l:

$$s_l \geq \lambda s_i + (1-\lambda) s_j, \tag{20}$$

which establishes Eq. 13.

To further analyze the relationship between s_i and s_j, we first rewrite their difference as:

$$s_i - s_j = q^\top (e_i - e_j). \tag{21}$$

We then apply Lemma 2 to upper bound this term:

$$s_i - s_j \leq \|q\| \cdot \|e_i - e_j\| = \|e_i - e_j\|. \tag{22}$$

Next, we explicitly compute the norm of the difference between the two embeddings:

$$\|e_i - e_j\|^2 = \|e_i\|^2 + \|e_j\|^2 - 2 e_i^\top e_j = 2 - 2\rho_{ij}. \tag{23}$$

Substituting this result back yields a lower bound for s_j:

$$s_j \geq s_i - \sqrt{2 - 2\rho_{ij}}, \tag{24}$$

which establishes Eq. 14.

Finally, by substituting Eq. 14 into Eq. 13, we derive:

$$s_l \geq \lambda s_i + (1-\lambda)\left(s_i - \sqrt{2 - 2\rho_{ij}}\right). \tag{25}$$

After simplification, we obtain the desired result:

$$s_l \geq s_i - (1-\lambda)\sqrt{2 - 2\rho_{ij}}, \tag{26}$$

which completes the proof of Eq. 15.

B.2 Theoretical Analysis of MemForest

In this section, we provide a detailed theoretical proof for MemForest.

Proof:

We analyze the memory merging strategy of MemForest. Specifically, MemForest compresses the historical memory set $H = \{m_i\}\,|_{i=1}^{N}$ into a compact set $H' = \{m'_j\}\,|_{j=1}^{M}$, where $M < N$.

For each EventTree C_k, as shown in Eq. 3, the partitioning process jointly considers global semantic similarity and local temporal continuity. Therefore, at the initialization stage, the similarity between any two nodes m_i and m_j satisfies $\rho_{ij} \geq \rho_{\min}^{(0)}$, where $\rho_{\min}^{(0)}$ denotes the minimum similarity threshold under the initial partitioning. This threshold is typically maintained at a relatively high level.

During progressive merging, the minimum similarity dynamically evolves as new nodes are introduced. Let $\rho_{\min}^{(t)}$ denote the minimum similarity after the t-th merge, then $\rho_{\min}^{(t)} \leq \rho_{\min}^{(t-1)}$. As indicated in Eq. 15, MemForest always prioritizes merging the node pair with the highest similarity at each step, thereby keeping $\rho_{\min}$ as large as possible after each merge and effectively mitigating the accumulation of semantic deviation.

According to Eq. 15, at the t-th merging step, the similarity loss incurred by a single merge is bounded by:

$$s_i - s_l = \Delta s^{(t)} \leq (1-\lambda)\sqrt{2 - 2\rho_{\min}^{(t)}}. \tag{27}$$

For the k-th EventTree C_k, let n_k and n'_k denote the numbers of memory nodes before and after compression, respectively. The total number of merging operations is:

$$n_k - n'_k. \tag{28}$$

Denoting the similarity loss at each step as $\Delta s^{(t)}$, the total similarity loss over the entire compression process can be written as:

$$\Delta S_k = \sum_{t=1}^{n_k - n'_k} \Delta s^{(t)}. \tag{29}$$

Substituting Eq. 27 into the above equation, we obtain:

$$\Delta S_k \leq \sum_{t=1}^{n_k - n'_k} (1-\lambda)\sqrt{2 - 2\rho_{\min}^{(t)}}. \tag{30}$$

Since $\rho_{\min}^{(t)}$ is monotonically non-increasing, we obtain the worst-case bound:

$$\Delta S_k \leq (n_k - n'_k)(1-\lambda)\sqrt{2 - 2\rho_{\min}^{(f)}}, \tag{31}$$

where $\rho_{\min}^{(f)}$ denotes the minimum similarity after the final merging step.

Summing over all EventTrees yields the overall loss bound:

$$\Delta S \leq (N - M)(1-\lambda)\sqrt{2 - 2\rho_{\min}^{(f)}}. \tag{32}$$

Thus, the overall query relevance of the compressed set H' satisfies:

$$\sum_{m'_j \in H'} s'_j \geq \sum_{m_i \in H} s_i - (N - M)(1-\lambda)\sqrt{2 - 2\rho_{\min}^{(f)}}. \tag{33}$$

This shows that the compression loss depends on the compression scale $(N - M)$ and the minimum similarity within EventTrees. Since MemForest prioritizes merging highly similar nodes, $\rho_{\min}^{(t)}$ can be consistently maintained at a relatively high level, effectively controlling error accumulation. As a result, MemForest maintains high query relevance even under high compression ratios.

C Memory Node Merging Prompt

The prompt used in our memory node merging process is as follows:

Prompt Used in Memory Node Merging

You are an intelligent memory summarization assistant. Please extract the core information from the following two memory entries and their timestamps, and generate a concise and coherent summary paragraph. If the memories contain relative time expressions (e.g., "last June," "two months ago"), convert them into specific dates based on the corresponding timestamps (for example, if a memory mentions "last June" and the timestamp is February 2023, the actual time should be June 2022). The summary length should be kept within the combined length of the two original memories as much as possible. Output only the final summarized result, without any explanations, analysis, reasoning steps, or additional content.

Memory 1: {}, Timestamp 1: {}

Memory 2: {}, Timestamp 2: {}

D Additional Experiments

D.1 Ablation Study under Different Ratios

Table 5: **Impact of different EventTree partitioning strategies at 30% compression rate.**

Method	LoCoMo	LongMemEval	M3-Bench-web	Avg.	%
Baseline	60.2	55.2	47.9	54.4	100.0%
w/o $g^j_{c_i}$	56.7	51.4	46.0	51.4	94.5%
w/o $l^j_{c_i}$	58.1	53.0	47.0	52.7	96.9%
Ours	**58.5**	**54.2**	**47.2**	**53.3**	**98.0%**

Table 6: **Impact of different memory node merging strategies at 30% compression rate.**

Method	LoCoMo	LongMemEval	M3-Bench-web	Avg.	%
Baseline	60.2	55.2	47.9	54.4	100.0%
Random	57.0	51.6	46.4	51.7	95.0%
Minimum	54.6	51.0	45.9	50.5	92.8%
Ours	**58.5**	**54.2**	**47.2**	**53.3**	**98.0%**

Table 7: **Impact of different EventTree partitioning strategies at 70% compression rate.**

Method	LoCoMo	LongMemEval	M3-Bench-web	Avg.	%
Baseline	60.2	55.2	47.9	54.4	100.0%
w/o $g^j_{c_i}$	51.4	49.6	45.1	48.7	89.5%
w/o $l^j_{c_i}$	53.6	51.8	46.0	50.5	92.8%
Ours	**54.4**	**52.2**	**46.3**	**51.0**	**93.8%**

Table 8: **Impact of different memory node merging strategies at 70% compression rate.**

Method	LoCoMo	LongMemEval	M3-Bench-web	Avg.	%
Baseline	60.2	55.2	47.9	54.4	100.0%
Random	51.5	50.8	45.3	49.2	90.4%
Minimum	49.8	48.0	44.2	47.3	87.0%
Ours	**54.4**	**52.2**	**46.3**	**51.0**	**93.8%**

As shown in Tables 5-8, when compressing **30%** and **70%** of historical memory, both the global similarity score $g^j_{c_i}$ and the local continuity score $l^j_{c_i}$ significantly improve performance. In addition, employing the other two merging strategies results in a notable performance drop, which is consistent with the previous experiments.

D.2 Other Ablation Study

Table 9: **Effect of node degree at different compression ratios.**

Method	LoCoMo	LongMemEval	M3-Bench-web	Avg.	%
Baseline	60.2	55.2	47.9	54.4	100%
w/o deg(·) (↓ 30%)	58.1	53.6	47.2	53.0	97.4%
w/ deg(·) (↓ 30%)	**58.5**	**54.2**	**47.2**	**53.3**	**98.0%**
w/o deg(·) (↓ 50%)	57.1	51.4	**46.6**	51.7	95.0%
w/ deg(·) (↓ 50%)	**58.4**	**51.8**	46.5	**52.2**	**96.0%**
w/o deg(·) (↓ 70%)	53.6	50.4	46.0	50.0	91.9%
w/ deg(·) (↓ 70%)	**54.4**	**52.2**	**46.3**	**51.0**	**93.8%**

As shown in Table 9, we conducted ablation experiments on the effect of node degree $\deg(\cdot)$ under different compression ratios. Across three representative benchmarks, incorporating node degree consistently improves performance compared to not using it, indicating that in the maximum spanning tree, nodes with higher degrees are typically core nodes of the event, carry more critical information, and should not be merged prematurely.

Table 10: **Performance comparison of different baselines on the LoCoMo benchmark in terms of B1 and F1 under varying compression ratios.** Blue denotes pruning methods, while Red denotes merging methods. For each ratio, the top two results are highlighted in **bold black font**.

Category	SH		MH		TR		OD		**All**	
Method	B1	F1	B1	F1	B1	F1	B1	F1	B1	F1
Mem0	36.6	46.6	22.6	32.3	40.6	49.9	14.5	20.3	33.5	42.0
+ AGPR	36.3	43.9	21.6	32.0	39.7	48.7	13.5	19.9	32.9	41.4
Historical memory compression ratio (↓ 30%)										
+ Random Pruning	29.5	36.3	20.6	30.7	34.6	40.9	14.5	17.5	28.0	35.1
+ KMeans	33.4	40.9	21.4	32.0	39.5	46.7	14.6	20.1	31.3	39.3
+ DART	34.4	41.9	22.1	32.1	35.9	43.4	13.8	19.4	31.2	39.0
+ Random Merging	32.7	39.9	20.0	29.7	35.6	46.4	16.3	21.7	30.0	38.3
+ ToMe	33.4	41.4	20.9	31.2	37.4	47.2	15.9	20.7	30.9	39.5
+ MemForest	33.8	41.6	22.6	33.8	37.1	46.7	18.0	23.4	**31.5**	**40.1**
+ MemForest + AGPR	34.1	41.8	23.0	34.4	38.3	48.2	17.3	23.3	**31.9**	**40.6**
Historical memory compression ratio (↓ 50%)										
+ Random Pruning	23.2	28.4	16.5	25.5	27.2	32.5	14.1	19.3	22.2	28.1
+ KMeans	28.1	34.1	16.9	25.9	30.7	36.4	13.9	17.6	25.7	32.0
+ DART	31.8	37.9	21.5	30.8	34.1	40.6	15.3	20.8	29.3	36.1
+ Random Merging	28.3	35.5	21.4	30.8	35.6	45.4	15.0	19.8	27.7	35.8
+ ToMe	31.1	38.3	20.8	30.4	35.0	45.4	15.2	19.7	29.1	37.2
+ MemForest	32.6	39.7	21.9	31.1	36.7	45.9	16.2	21.6	**30.5**	**38.3**
+ MemForest + AGPR	32.6	40.0	23.9	33.7	36.4	45.7	15.4	20.8	**30.7**	**38.8**
Historical memory compression ratio (↓ 70%)										
+ Random Pruning	18.0	22.6	13.9	21.2	21.4	25.1	11.2	15.1	17.5	22.4
+ KMeans	20.7	25.7	14.4	23.3	23.7	27.5	14.9	20.7	19.9	25.3
+ DART	25.6	31.0	18.9	27.6	28.4	34.1	13.6	19.1	24.2	30.3
+ Random Merging	27.0	33.5	19.6	29.6	36.8	45.4	17.4	22.3	27.1	34.6
+ ToMe	27.7	34.7	20.2	29.3	34.4	44.5	12.8	17.6	26.8	34.7
+ MemForest	29.3	35.8	23.6	32.9	34.9	42.5	14.2	20.1	**28.4**	**35.7**
+ MemForest + AGPR	29.6	36.2	23.3	32.6	35.1	43.0	15.5	21.6	**28.7**	**36.0**

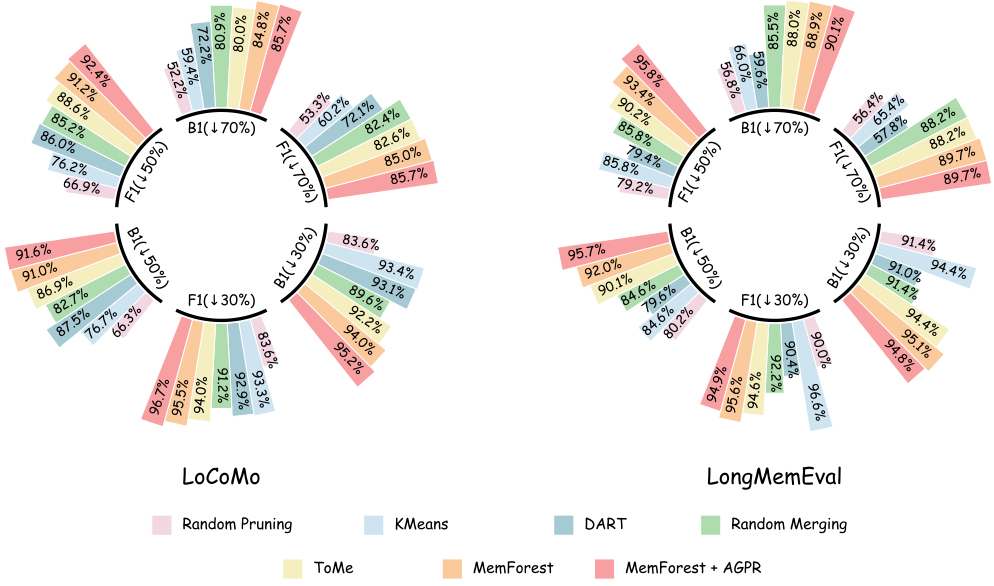


Figure 4: **Retention of B1 and F1 scores.** On two benchmarks, the percentages of B1 and F1 retained relative to the original performance across methods under varying compression rates.

Table 11: **Performance comparison of different baselines on the LongMemEval benchmark in terms of B1 and F1 under varying compression ratios.** Blue denotes pruning methods, while Red denotes merging methods. For each ratio, the top two results are highlighted in **bold black font**.

Category	SU		SA		SP		MS		KU		TR		All	
Method	B1	F1	B1	F1	B1	F1	B1	F1	B1	F1	B1	F1	B1	F1
Mem0	64.1	72.6	15.1	15.5	0.1	7.6	32.1	38.2	39.7	51.1	26.3	38.9	32.4	40.8
+ AGPR	62.7	71.8	14.9	15.3	0.1	8.0	30.4	35.7	40.5	52.4	26.5	38.9	31.9	40.3
Historical memory compression ratio (↓ 30%)														
+ Random Pruning	58.7	67.2	14.3	13.3	0.1	7.8	27.2	32.0	35.3	44.7	26.3	37.2	29.6	36.7
+ KMeans	60.1	59.4	13.8	14.4	0.1	7.7	29.4	35.7	39.6	51.4	24.8	38.0	30.6	**39.4**
+ DART	57.2	65.8	14.7	16.2	0.0	6.1	27.4	31.3	39.5	50.2	24.0	35.3	29.5	36.9
+ Random Merging	60.0	69.1	12.9	14.1	0.0	6.9	26.1	30.4	37.3	48.9	26.3	38.4	29.6	37.6
+ ToMe	59.3	68.6	14.8	15.8	0.0	7.1	29.7	34.2	37.0	47.9	26.3	38.3	30.6	38.6
+ MemForest	58.9	69.1	13.3	13.6	0.0	6.0	30.8	35.5	38.4	49.5	25.8	38.7	**30.8**	**39.0**
+ MemForest + AGPR	60.4	69.0	13.2	13.3	0.0	6.4	30.1	35.2	38.7	50.2	25.2	37.5	**30.7**	38.7
Historical memory compression ratio (↓ 50%)														
+ Random Pruning	49.7	57.5	12.6	13.9	0.0	6.9	24.2	25.0	30.6	39.5	24.2	35.5	26.0	32.3
+ KMeans	53.1	61.8	12.1	13.3	0.0	6.8	24.9	28.3	34.9	46.3	24.6	36.3	27.4	35.0
+ DART	47.4	54.7	11.7	11.1	0.1	7.7	26.2	30.2	30.3	39.8	23.3	33.1	25.8	32.4
+ Random Merging	55.2	63.7	14.1	15.2	0.1	9.2	25.2	29.4	30.6	41.4	24.8	35.9	27.4	35.0
+ ToMe	56.5	66.5	13.2	13.4	0.0	7.2	29.6	34.1	31.8	41.8	26.3	37.3	29.2	36.8
+ MemForest	59.8	69.5	13.2	14.6	0.0	7.0	28.5	33.1	38.3	49.2	24.1	36.8	**29.8**	**38.1**
+ MemForest + AGPR	60.6	70.2	15.2	16.4	0.0	7.6	28.7	33.3	38.8	49.0	26.6	39.4	**31.0**	**39.1**
Historical memory compression ratio (↓ 70%)														
+ Random Pruning	30.3	36.1	7.0	7.6	0.1	8.0	15.6	14.0	20.8	27.0	22.3	32.6	18.4	23.0
+ KMeans	44.0	51.0	9.6	8.0	0.0	6.6	15.0	15.3	29.0	37.4	21.3	31.5	21.4	26.7
+ DART	28.4	32.7	10.1	9.3	0.1	5.4	20.7	21.9	22.3	30.5	19.4	26.7	19.3	23.6
+ Random Merging	54.1	65.2	17.2	19.3	0.1	8.3	25.3	30.6	32.3	43.8	24.0	34.7	27.7	36.0
+ ToMe	55.6	64.3	15.6	16.2	0.0	6.1	27.4	31.3	29.5	40.3	26.6	38.2	28.5	36.0
+ MemForest	54.2	63.9	16.1	17.1	0.0	7.3	28.5	32.8	34.7	44.0	24.2	36.4	**28.8**	**36.6**
+ MemForest + AGPR	55.7	65.2	16.6	18.4	0.0	6.7	28.4	33.0	34.2	43.6	25.0	35.5	**29.2**	**36.6**

D.3 Other Evaluation Metrics

Furthermore, for the Mem0 framework, we evaluated the **B1** and **F1** scores on two representative benchmarks: LoCoMo and LongMemEval. As shown in Tables 10–11 and Figure 4, our method consistently outperforms other approaches across both benchmarks, further demonstrating the effectiveness of MemForest.

D.4 Parameter Sensitivity Analysis

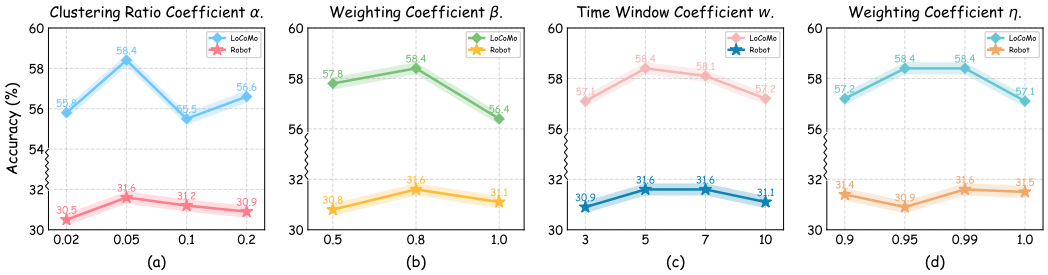


Figure 5: **Impact of Parameter Variations on Merging Performance.**

To analyze the sensitivity of different parameters, we conduct experiments on the LoCoMo and M3-Bench-robot datasets under a compression ratio of **50%**.

Clustering Ratio Coefficient α. As shown in Fig. 5(a), when the clustering ratio coefficient α is set to **0.05**, the performance on both benchmarks reaches its optimum. When α is either too high or too low, the performance degrades. This indicates that when the number of EventTrees is too small, excessive unrelated

memories are grouped within a single EventTree; whereas when the number is too large, memory nodes from the same event are overly fragmented, thereby harming overall performance.

Weighting Coefficient β. As shown in Fig. 5(b), when the weighting coefficient β is set to **0.8**, the performance on both benchmarks reaches its optimum. When β is too large, the effect of local temporal continuity is weakened; whereas when β is too small, temporal continuity becomes dominant, leading to a suppression of global semantic similarity, thereby degrading the overall performance.

Time Window Coefficient w. As shown in Fig. 5(c), when the time window coefficient w is set to **5**, the performance on both benchmarks reaches its optimum. When w is too large, each cluster contains too many neighboring memory nodes, which weakens the aggregation of semantically similar nodes; whereas when w is too small, the number of neighboring nodes is insufficient to provide adequate local temporal information, thereby affecting the overall performance.

Weighting Coefficient η. As shown in Fig. 5(d), the performance on both benchmarks reaches its optimum when the weighting coefficient η is set to **0.99**. When η is too small, the degree information dominates, which suppresses the merging of semantically similar nodes; whereas when η is set to **1**, the importance of node degrees is ignored, potentially causing core nodes to be merged prematurely.

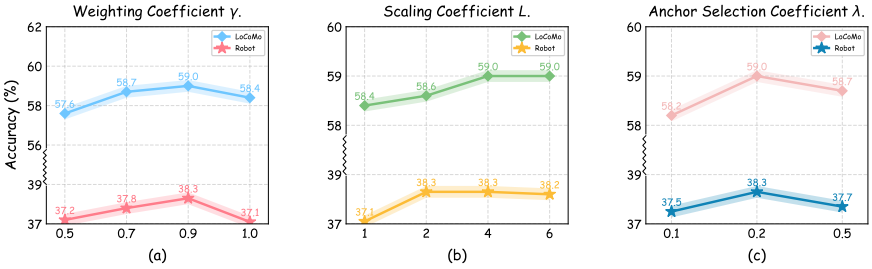


Figure 6: **Impact of Parameter Variations on Memory Retrieval.**

Weighting Coefficient γ. As shown in Fig. 6(a), when the weighting coefficient γ is set to **0.9**, the performance on both benchmarks reaches its optimum. When γ is too low, the effect of semantic similarity on retrieval is weakened, leading to degraded performance; whereas when $\gamma = \mathbf{1}$, retrieval relies solely on semantic similarity, ignoring the neighborhood information of key memory nodes, which results in performance degradation.

Scaling Coefficient L. As shown in Fig. 6(b), when the scaling coefficient L is set to **4**, the performance on both benchmarks reaches its optimum. When L is too small, the number of recalled candidate memory nodes is insufficient to provide adequate temporal neighborhood compensation, thereby affecting retrieval performance; whereas when L is greater than **4**, the performance remains almost unchanged, as the number of candidate memory nodes is sufficient to fully cover the neighborhood of key memory nodes.

Anchor Selection Coefficient λ. As shown in Fig. 6(c), when the anchor selection coefficient λ is set to **0.2**, the performance on both benchmarks reaches its optimum. When λ is too low, the number of anchors is insufficient, resulting in inadequate propagation of neighborhood information and preventing some relevant memory nodes from being effectively retrieved; whereas when λ is too high, excessive anchors introduce redundant information and additional noise, which negatively affects retrieval performance.

D.5 Memory Node Merging Cost Analysis

Regarding the cost of memory node merging, each merge only requires a short prompt and two memory entries as input, producing a single summarized memory, resulting in very low overall cost. When using GPT-4o-mini to merge memory nodes and compress **50%** of the historical memory, the cost is only about **$0.1** for the LoCoMo benchmark (≈100MB, 6,000 memory nodes) or roughly two hours of streaming video, making the practical overhead negligible. Moreover, this process can also be implemented with locally deployed open-source models.

D.6 Merging Performance across Different Models

We evaluate Qwen2.5-7B-Instruct [34], GPT-4o-mini, and GPT-5.2 under a **50%** compression ratio on the LoCoMo benchmark. As shown in Table 12, GPT-4o-mini and GPT-5.2 achieve comparable merging performance when used as backbone models, suggesting that once the model size reaches a certain scale, the performance tends to saturate with limited room for further improvement. In contrast, Qwen2.5-7B-Instruct performs worse,

likely due to its smaller parameter size. This observation also raises an interesting question: can a dedicated lightweight model be trained specifically for memory merging, and could it achieve performance comparable to or even surpass that of models like GPT-4o-mini?

Table 12: **Merging performance of different models.**

Method	LoCoMo	%
Baseline	60.2	100.0%
Qwen2.5-7B-Instruct	52.9	87.9%
GPT-4o-mini	**58.4**	**97.0%**
GPT-5.2	58.1	96.5%

D.7 Detailed time efficiency results

Table 13: ***T-Time(s)* across the LoCoMo, LongMemEval, and PersonaMem benchmarks.**

Dataset	LoCoMo	LongMemEval	PersonaMem
Baseline	0.48	8.30	4.90
Baseline + AGPR	0.48	8.31	4.90
Baseline + MemForest + AGPR ($\downarrow$ 30%)	0.30	6.04	3.56
Baseline + MemForest + AGPR ($\downarrow$ 50%)	0.24	4.32	2.78
Baseline + MemForest + AGPR ($\downarrow$ 70%)	0.16	2.70	1.34

Table 14: ***Avg-R(×), R-Time(s), and T-Time(s)* across M3-Bench-robot and M3-Bench-web.**

Dataset	M3-Bench-robot			M3-Bench-web		
	Avg-R	R-Time	T-Time	Avg-R	R-Time	T-Time
Baseline	3.10	0.32	0.99	2.32	0.31	0.72
Baseline + MemForest + AGPR	2.52	0.32	0.81	2.12	0.32	0.68
Baseline + MemForest + AGPR ($\downarrow$ 30%)	2.60	0.21	0.55	2.03	0.18	0.37
Baseline + MemForest + AGPR ($\downarrow$ 50%)	2.59	0.16	0.41	2.20	0.16	0.35
Baseline + MemForest + AGPR ($\downarrow$ 70%)	2.62	0.12	0.31	2.02	0.12	0.24

Tables 13–14 present the detailed results in Fig. 3, showing that our method achieves significant speedup.

E Case Study

EventTree Partitioning Process. The following shows an initial EventTree obtained during the EventTree partitioning process on the LoCoMo benchmark. The memory nodes within this EventTree primarily revolve around *"transgender and community support"*. It can be observed that some memory nodes, though not temporally adjacent, exhibit high semantic similarity (e.g., nodes 1, 9, and 10, *"related to community support"*), while other nodes, which are temporally adjacent but have lower semantic similarity with most memory nodes (e.g., nodes 7 and 8, *"related to mentoring in the community"*), are also grouped into the same EventTree. This effectively integrates the global semantic similarity of memory events with their local temporal continuity.

An Example of an Initial EventTree Obtained from Partitioning

Memory 1: *Values inclusivity and support.*

Timestamp 1: *1:14 pm on 25 May, 2023*

Memory 2: *Finds participating in charity events rewarding.*

Timestamp 2: *1:14 pm on 25 May, 2023*

Memory 3: *Aims to build a strong, supportive community of hope.*

Timestamp 3: *7:55 pm on 9 June, 2023*

Memory 4: *Believes in building a more inclusive and understanding world.*

Timestamp 4: *7:55 pm on 9 June, 2023*

Memory 5: *Excited to meet other people in the community.*
Timestamp 5: *1:36 pm on 3 July, 2023*

Memory 6: *Values the importance of fighting for trans rights and spreading awareness.*
Timestamp 6: *4:33 pm on 12 July, 2023*

Memory 7: *Has a mentee.*
Timestamp 7: *2:31 pm on 17 July, 2023*

Memory 8: *Mentors a transgender teen.*
Timestamp 8: *2:31 pm on 17 July, 2023*

Memory 9: *Passionate about rights and community support.*
Timestamp 9: *8:56 pm on 20 July, 2023*

Memory 10: *User is passionate about rights and community support.*
Timestamp 10: *8:56 pm on 20 July, 2023*

Memory 11: *The group has regular meetings and plans events and campaigns.*
Timestamp 11: *8:56 pm on 20 July, 2023*

Memory 12: *Passionate about helping people and making a positive impact.*
Timestamp 12: *8:56 pm on 20 July, 2023*

Memory 13: *Believes in the fight for equality and inclusivity.*
Timestamp 13: *8:56 pm on 20 July, 2023*

Memory 14: *Values standing up for equality.*
Timestamp 14: *2:24 pm on 14 August, 2023*

Memory 15: *Wants to spread understanding and acceptance.*
Timestamp 15: *1:33 pm on 25 August, 2023*

Memory 16: *Shared personal story with young people.*
Timestamp 16: *3:19 pm on 28 August, 2023*

Memory 17: *Proud of her identity.*
Timestamp 17: *12:09 am on 13 September, 2023*

Memory Node Merging Process. As shown below, we present an example from the M3-Bench-robot benchmark during the memory node merging process. The two nodes to be merged are highly related, and the resulting merged memory node fully preserves the original information while significantly improving storage efficiency and enhancing subsequent retrieval efficiency.

An Example of Merging Two Memory Nodes

Memory 1: *The task involves organizing and preparing documents, possibly for a presentation or application.*
Timestamp 1: *Clip 6*

Memory 2: *The task being performed involves document preparation, possibly for an application, report, or presentation.*
Timestamp 2: *Clip 6*

Merged memory: *The task involves organizing and preparing documents for a presentation, application, or report.*
Timestamp: *Clip 6*

F Limitations

We conducted experiments on both the high-redundancy multimodal memory framework M3-Agent and the low-redundancy textual memory framework Mem0. The results indicate that in Mem0, which has relatively low redundancy, performance degradation becomes pronounced at higher compression rates. For instance, when **70%** of the historical memory is compressed, the accuracy drops to only **93.3%** of that in the uncompressed setting. In the future, specialized compression methods could be designed for such low-redundancy memory frameworks to improve performance under high compression ratios.

G Future Work

We propose the following potential research directions for future work:

- MemForest is a general-purpose memory compression framework. Future work could explore designing specialized compression methods for low-redundancy memory frameworks like Mem0, in order to better preserve performance under high compression rates.
- As discussed in Section D.6, it may be possible to train a dedicated small memory-merging model that achieves, or even surpasses, the memory node merging performance of large models such as GPT-4o-mini.
- There is also room for optimization in memory retrieval mechanisms. Most current methods rely on selecting the top k most similar memory nodes, which involves only limited embedding similarity computations and can leverage vector database acceleration, typically resulting in retrieval times under one second. In contrast, some newer approaches, such as xMemory [15], require redesigning the memory generation framework, are incompatible with existing frameworks, and perform extensive embedding similarity computations during retrieval. They also require reconsideration of memory structure and cannot leverage vector database acceleration, leading to retrieval times that are tens of times longer. Therefore, future work could explore retrieval mechanisms that minimize embedding similarity computations while achieving a better balance between retrieval speed and accuracy.
- In terms of safety, it may be possible to perform a holistic analysis of the historical memory set to identify conflicts or potentially corrupted entries, and take appropriate measures to correct or remove them, thereby enhancing the robustness and reliability of the memory system.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 introduction accurately reflect the scope of our model's contributions; this is the first general-purpose memory compression framework and demonstrates strong performance.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of our method in Appendix F of the paper.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: We provide detailed theoretical proofs in Section 4 and Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.

- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide all hyperparameters and settings in Section 5 of the paper, and all experiments were repeated three times. We will release all data and code in the future.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We have provided the code in the supplementary material, and we commit to releasing all code and data in the future.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).

- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All datasets we used follow the testing protocols of Mem0 and M3-Agent. We will release all data and code in the future. Additionally, detailed information about the datasets and baselines is provided in Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Our main experiments and ablation studies report results averaged over three runs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide detailed information in Section 5 and Appendix A: experiments were conducted using dual A100 GPUs, and the APIs used include the GPT-4o series and text-embedding models.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: All aspects comply with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: Our work has no societal impact.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our paper does not involve such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have properly used all models and datasets in compliance with their respective terms. For details on the baselines and datasets, please refer to Appendix A.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The details of our method are provided in Section 3, the code is included in the supplementary materials, and we will release everything in the future.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: Our paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Our paper poses no risks and does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: The choice of which large language model is used for memory node merging is detailed in Section 5.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.