EMA: An Episodic Memory Agent for Efficient and Selective Memory

Hongyi Lan[1], Jiaqi Song[1,2], Zhengjia Zhong[1], Hui Li[1],
Hong Liu[1], Xianming Lin[1]*, Rongrong Ji[1,3]
[1]Key Laboratory of Multimedia Trusted Perception and Efficient Computing Ministry of Education of China, Xiamen University,
[2]Institute of Artificial Intelligence, Xiamen University,
[3]Sino-Russian Research Center for Digital Economy
{lanhongyi,songjjjqqq,zhongplusplus}@stu.xmu.edu.cn,
{hui,hlynn,linxm,rrji}@xmu.edu.cn

Abstract

Large Language Models (LLMs) demonstrate strong generation and reasoning abilities, but they still face challenges in long-term memory retention and multi-turn conversational consistency. Existing memory-augmented methods often incorporate full dialog histories without filtering, resulting in information redundancy and inference latency. Inspired by the episodic memory mechanism in human cognition, we abstract conversational context into Episodic Memory Units (EMUs). We then propose a comprehensive framework, Episodic Memory Agent (EMA), along with a filtering decision module called MemDecider. Specifically, EMA organizes and retrieves EMUs to support response generation, while MemDecider filters information to reduce noise and improve overall performance. Experiments on two widely-used benchmarks show that EMA maintains competitive performance, and integrating MemDecider into other methods reduces their token consumption by an average of 11.48% while effectively improving the overall performance. Code is available at https://github.com/Hongyi4221/EMA.

1 Introduction

Large Language Models (LLMs) have advanced significantly; yet, limited long-term memory and conversational consistency remain critical bottlenecks for personalized and history-aware applications (Yuan et al., 2025; Zhang et al., 2025; Hu et al., 2025). Although extending the context window increases the effective receptive scope of the model (Xu et al., 2025b; Wu et al., 2024; Su et al., 2024; Peng et al., 2024), research indicates that this does not fundamentally resolve the underlying issues (Li et al., 2024). This limitation is particularly evident in multi-turn dialogue, where models usually forget early details or suffer from interference, degrading coherence and personalization (Laban et al., 2025). Consequently, LLMs require efficient memory mechanisms for selective information retention to ensure robust conversational consistency.

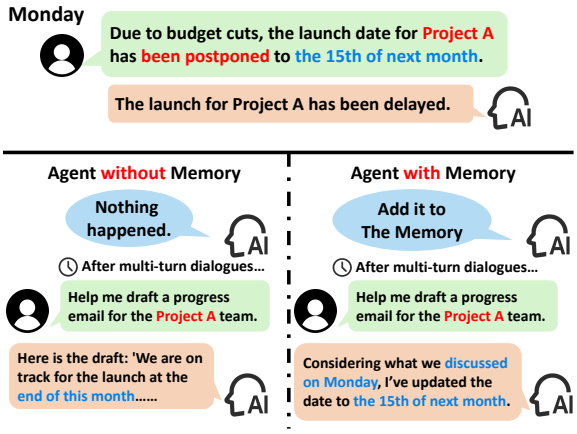

Figure 1: An example of a conversational scenario, comparing LLMs with and without AI Memory.

Motivated by this need, the research area of "AI Memory " has emerged (Wu et al., 2025). Unlike long-context methods (Liu et al., 2025) that merely expand the model's view or retrieval-augmented generation (RAG) approaches (Fan et al., 2024) that rely on a static knowledge base, AI Memory can dynamically extract, organize, and store key information from ongoing interactions (Yuan et al., 2025; Zhang et al., 2025). As shown in Figure 1, AI Memory maintains consistency and continuity in multi-turn dialogue, whereas LLMs without AI Memory tend to forget earlier information.

Recent research introduces external memory modules or frameworks to enhance AI agent memory (Tan et al., 2025; Zhong et al., 2024). These approaches are generally categorized into fine-tuning specialized memory components (Yu et al., 2025; Dai et al., 2025; Jin et al., 2025) or designing memory structures via prompt engineering and backbone capabilities (Tan et al., 2025; Gutiérrez et al., 2025; Zhong et al., 2024). However, existing methods typically feed entire dialogue histories into

*Corresponding author

memory systems without importance filtering, resulting in redundancy and inefficiency.

Human episodic memory selectively prioritizes key experiences, achieving greater parsimony and efficiency (Miller, 1956). Studies on episodic and working memory (Cowan, 2012; Tulving, 2002) suggest that humans retain contextually relevant events while filtering out irrelevant details. Inspired by this, we introduce a new Episodic Memory Unit (EMU) that abstracts conversational context. Then, we propose MemDecider, an upstream decision module that controls input retention. Thus, by filtering critical information, MemDecider maintains high performance while significantly reducing inference latency and token consumption. Notably, as a plug-and-play module, MemDecider can be seamlessly integrated into most AI Memory systems. Finally, we abstract the filtered inputs into EMUs for storage and management, forming a comprehensive framework Episodic Memory Agent, named EMA.

We evaluate EMA on two classical benchmarks, *i.e.,* Locomo (Maharana et al., 2024) and LongMT-Bench+ (Pan et al., 2025). The results show that EMA achieves competitive performance while reducing token usage by an average of 11.48%, thereby lowering inference latency and API invocation costs. In sum, our contributions are as follows.

- **A Unified Memory Framework.** We abstract memory content into Episodic Memory Units to build a complete framework, EMA. This design aims to achieve competitive performance while enhancing efficiency.
- **A Decision Module.** We propose MemDecider, an upstream and plug-and-play module that filters memory admission based on information importance. To our knowledge, this is the first front-end module to assess content significance for guiding memory generation.
- **Performance and Efficiency.** Experimental results demonstrate that EMA achieves strong performance on two benchmarks with lower token consumption. Furthermore, MemDecider enhances the efficiency of other existing methods while maintaining their performance levels.

2 Preliminary

Our approach is inspired by episodic memory (Tulving, 2002) and seeks to apply this theory from human cognition to AI Memory in order to enhance memory capabilities.

Episodic memory refers to a form of memory that records an individual's experienced events, with its core being the holistic binding of the event and context. An episodic memory typically includes elements such as event content, time and location, participants, and subjective importance. Based on this, we identify and extract dialogue components aligned with the core concepts of episodic memory and organize them into episodic memory units, termed EMUs.

For a given dialogue sequence $D = u_1, \ldots, u_T$, segments that meet certain triggering conditions are abstracted as an EMU $E = e_i{}_{i=1}^{N}$, where EMU is defined as follows:

$$e_i = \langle \mathrm{id}_i,\ \tau_i,\ P_i,\ \phi_i,\ C_i,\ s_i,\ c_i,\ L_i,\ x_i \rangle,$$

where id_i denotes the EMU ID; τ_i represents the time of the event; P_i is the set of participants; ϕ_i is an event-level summary of the memory; C_i is a set of semantic or contextual labels used for retrieval; $s_i \in [0, 1]$ indicates the importance of the memory; $c_i \in [0, 1]$ represents the memory's confidence; L_i denotes the set of links to related memories, which is used to connect different EMUs across events; and x_i denotes the original dialogue content.

3 Method

3.1 Overview of EMA

We propose a comprehensive memory management framework, Episodic Memory Agent, called EMA. EMA utilizes EMUs as the fundamental building blocks for external memory, supported by specific components for construction, retrieval, and response generation.

As illustrated in Figure 2, the framework comprises three core modules: (1) MemDecider ensures efficiency and accuracy by filtering critical dialogue content as raw data for EMUs. (2) Memory Manager serves as the central hub, transforming raw data into structured EMUs while managing and maintaining these units; (3) Caller handles retrieval and leverages EMUs to generate precise and consistent responses. Implementation details, including prompts and hyperparameter analysis, are provided in Appendix A and Appendix C.

3.2 MemDecider

A key limitation of existing methods is the indiscriminate passing of all content to downstream processes without importance filtering. In contrast, humans selectively encode key experiences

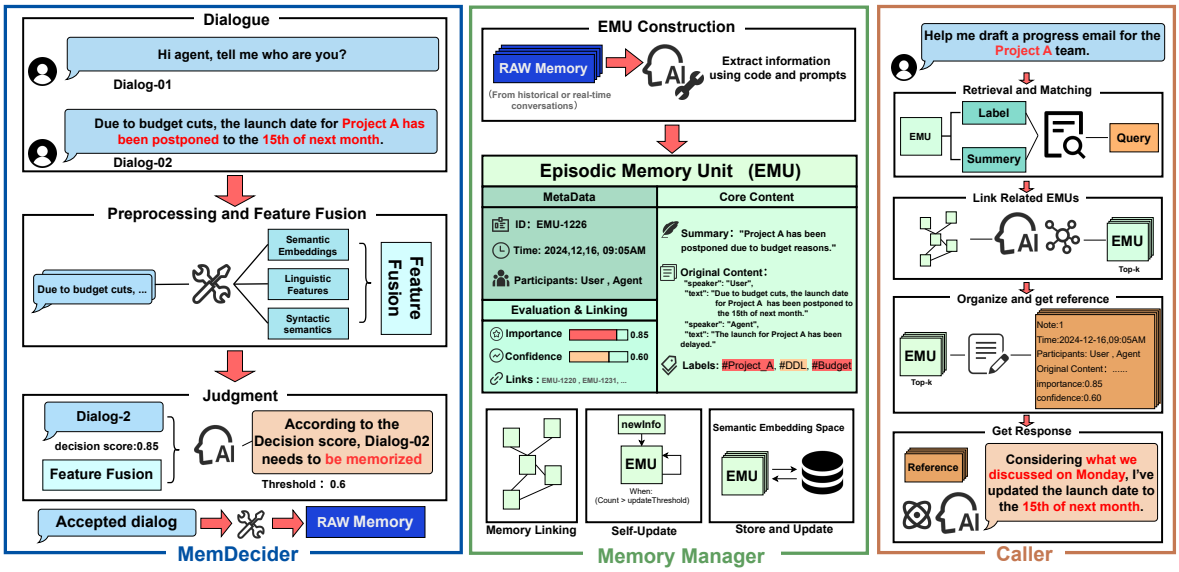

Figure 2: An overview of EMA. It consists of three key components: MemDecider evaluates and filters text importance; Memory Manager constructs and maintains EMUs; and Caller retrieves EMUs and generates responses.

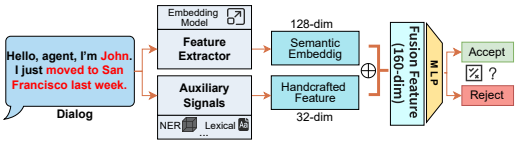

Figure 3: The architecture of MemDecider, which serves as a streamlined decision-making module.

to reduce cognitive load and optimize memory retrieval (Tulving, 2002). MemDecider acts as a front-end screening module that filters input before it enters the main framework. Briefly, it filters inputs to ensure that only significant content is memorized. Thus, this involves retaining essential information as raw data for constructing EMUs.

To maintain efficiency, we design MemDecider with a streamlined architecture as shown in Figure 3. MemDecider leverages both semantic embeddings and hand-crafted features. Where semantic embeddings capture deep contextual meaning, hand-crafted features characterize text structures and essential elements. These fused features are then fed into an MLP classifier to determine whether the input text should be retained in memory.

Semantic embeddings $\mathbf{E}$ are obtained via a pre-trained embedding model $\mathbf{M}$, which maps the raw text input C to a high-dimensional feature vector $\mathbf{x}$. Let d_e denote the embedding dimension; the semantic features are defined as:

$$\mathbf{E} = \text{MeanPool}(\mathcal{M}(C)) \in \mathbb{R}^{d_e}.$$

These embeddings capture abstract dialogue information, supporting EMU construction and linking while facilitating the subsequent retrieval process.

For the hand-crafted features, we categorize them into two parts: linguistic statistical features $\mathbf{L}$ and syntactic semantics $\mathbf{S}$. Linguistic statistical features $\mathbf{L}$ are related to sentence structure:

$$\mathbf{L} = [f_{\text{len}}, f_{\text{uniq}}, f_{\text{digit}}, f_{\text{punct}}, f_{\text{upper}}, f_{\text{img}}, f_{\text{q}}] \in \mathbb{R}^{d_s},$$

where f_{len} represents sentence length and f_{uniq} measures lexical diversity. f_{digit}, f_{punct}, and f_{upper} are the counts of digits, punctuation marks, and uppercase letters, respectively. f_{img} indicates the presence of multimodal elements, and f_{q} captures the frequency of question words or question marks.

Syntactic semantics $\mathbf{S}$ are derived from further sentence attributes, which are defined as:

$$\mathbf{S} = [f_{\text{person}}, f_{\text{gpe}}, f_{\text{time}}, f_{\text{noun}}, f_{\text{verb}}] \in \mathbb{R}^{d_l},$$

where f_{person}, f_{gpe}, and f_{time} correspond to the results of Named Entity Recognition for persons, geopolitical entities, and temporal expressions, respectively. While f_{noun} and f_{verb} represent noun and verb counts obtained from Part-of-Speech tagging, respectively.

Finally, the fused feature vector is constructed by concatenating all the features:

$$\mathbf{x} = \Phi(C) = [\mathbf{E}; \mathbf{L}; \mathbf{S}] \in \mathbb{R}^{d},$$

where $d = d_s + d_l + d_e$ denotes the dimensionality of the final input vector. This fused feature

$\mathbf{x}$ is fed into an MLP to generate a decision score. Moreover, we introduce a hyperparameter τ as the decision threshold; only content with a score exceeding τ is classified as significant information and is subsequently constructed into a structured EMU. A detailed analysis of τ is provided in Section 4.4.

3.3 Memory Manager

The Memory Manager serves as the central hub of EMA, responsible for transforming the critical information selected by MemDecider into structured EMUs. Its primary roles within the framework include the creation, storage, and maintenance of these units. Here, EMUs simulate human episodic memory by integrating multi-dimensional information, including event content, timestamps, participants, key tags, and semantic summaries. In addition, each EMU is assigned an "importance" score and a "confidence" score to serve as references for the response module. Specifically, the Memory Manager implements the following dynamic mechanisms.

Memory Linking. Each newly generated EMU is endowed with keywords and summaries via heuristic rules and prompt engineering. The Memory Manager links semantically related EMUs to form a structured memory network, thus enhancing retrieval accuracy and supporting complex multi-hop reasoning.

Self-Update. Following Memory Linking, relevant memories and events are updated as needed. For example, if a user mentions a plan to visit New York on Monday but cancels it on Wednesday, the corresponding EMUs will be updated to reflect the latest information. When new EMUs are semantically or temporally related to existing ones, the Memory Manager triggers updates to eliminate redundancy and maintain information accuracy.

EMUs Reorganization. As EMUs accumulate, EMA necessitates periodic reorganization to maintain retrieval efficiency. While manual intervention is possible, the Memory Manager features a dynamic triggering mechanism. In short, once the frequency of Memory Linking and Self-Update operations reaches a predefined threshold, it executes a global re-embedding and consolidation. This dynamic process, driven by code and prompt engineering, refines the memory network by eliminating redundancy and ensuring sustained high accuracy and response efficiency.

3.4 Caller

Caller module is responsible for retrieval and response generation within EMA. It enriches user queries via Named Entity Recognition (NER) and employs embedding-based retrieval to identify the top-k relevant EMUs. By interfacing with the Manager, Caller retrieves comprehensive metadata to reconstruct context, ultimately synthesizing an augmented prompt that drives the LLM to produce accurate and coherent responses. The detailed procedure is described below.

When a user input is received, the Caller first performs Named Entity Recognition (NER) to extract key semantic entities. These entities are concatenated with the original input to form a structured query, enhancing semantic specificity and retrieval relevance. Formally, let the input be I, and the extracted entities be $E_{\text{NER}}(I)$. The structured query is defined as

$$Q = I \oplus E_{\text{NER}}(I).$$

The structured query Q is then encoded by an embedding model to produce a query vector $v_q = E(Q)$. Based on this representation, the Caller retrieves the top-k most relevant EMUs from the memory collection by matching against their metadata, including label, summary, and link information. For each EMU m_i, its representation is defined as

$$v_{m_i} = E(m_i.\text{label} \oplus m_i.\text{summary} \oplus m_i.\text{links}).$$

After retrieval, Caller invokes the corresponding functions of Memory Manager to obtain complete information about the selected EMUs, including original content, timestamps, importance scores, confidence values, and other related metadata. During the generation stage, Caller produces the final response based on this information. The aggregated context is defined as

$$\begin{aligned} C_{\text{complete}} = \{&(m.\text{content}, m.\text{timestamp}, \\ &m.\text{importance}, m.\text{confidence}) \\ &\mid m \in M_{\text{retrieved}}\}. \end{aligned}$$

Finally, Caller integrates the structured query Q with the retrieved context to construct an augmented prompt, which is fed into the LLM to generate responses that are coherent, contextually grounded, and semantically consistent:

$$R = \text{LLM}(Q, C_{\text{complete}}).$$

3.5 Optimization

Within the EMA framework, only the MemDecider module undergoes training. We adopt a two-stage training strategy to combine the efficiency of supervised learning with the adaptive nature of reinforcement learning. We use 50% of the Locomo dataset (Maharana et al., 2024) for this process.

Stage 1: Supervised Initialization

We initialize the decision module to establish fundamental discriminative capability. Training samples are formatted as (c_i, y_i), where $y_i \in \{0, 1\}$ indicates the memory acceptance of content c_i. The module is optimized by minimizing the Binary Cross-Entropy loss, which is defined as:

$$\mathcal{L} = -\sum_i \left[y_i \log p_i + (1 - y_i) \log(1 - p_i) \right],$$

where p_i is the predicted probability of sample i.

Stage 2: Reinforcement Learning

Reinforcement learning (RL) is employed for this step optimization, with F1, ROUGE-L (Lin, 2004), BLEU (Papineni et al., 2002), and EM (Rajpurkar et al., 2016) serving as key performance metrics. To address the credit assignment problem in long-context tasks, we incorporate Reward Shaping (Ng et al., 1999) to integrate downstream feedback with step-wise feedback. The reward function R is formulated as a composite signal:

$$R_{qa} = w_1 \cdot (\text{F1}) + w_2 \cdot (\text{ROUGE-L}) + w_3 \cdot (\text{BLEU-1}) + w_4 \cdot (\text{EM}),$$

$$R_{total} = \alpha \cdot R_{qa} + \beta \cdot \sum_{t=1}^{T} r_{step,t},$$

where w_1, w_2, w_3, and w_4 denote the weight coefficients of each evaluation metric, respectively. R_{qa} is the global terminal reward derived from the composite signal R at completion, while $r_{step,t}$ denotes the intermediate reward at decoding step t. T is the sequence length, and α, β are scaling hyperparameters that balance global quality assessment and step-wise feedback for stable policy gradient updates. Specifically, $r_{step,t}$ assigns $+2.0$ to each correctly identified evidence item and -0.5 to redundant storage, promoting accurate and concise reasoning. Detailed analysis of these coefficients and hyperparameters is provided in Appendix C.

We adopt the REINFORCE algorithm (Schulman et al., 2017), modeling the decision module as a parameterized stochastic policy $\pi_\theta(a \mid s)$, where action $a \in \{0, 1\}$ determines whether the content is committed to memory.

To stabilize training and reduce gradient variance, we incorporate Advantage Normalization alongside Entropy Regularization to encourage exploration and prevent premature convergence. The policy parameters are optimized as follows:

$$\nabla_\theta J(\theta) \approx \sum_{t=1}^{T} \Big[(R_{total} - b) \nabla_\theta \log \pi_\theta(a_t \mid s_t) + \lambda \nabla_\theta H(\pi_\theta(s_t)) \Big],$$

where R_{total} is the cumulative reward, and b denotes the baseline for advantage normalization to reduce variance. $H(\pi_\theta(s_t))$ represents the Shannon entropy of the policy at step t used as a regularization term, with λ as the entropy regularization coefficient. To avoid symbolic conflict with the scaling factors in the reward function, the entropy weight is denoted as λ.

4 Experiments[1]

4.1 Dataset and Implementation Details

We follow the experimental setup and evaluation metrics in the MemoryOS work (Kang et al., 2025). We conduct extensive experiments on the Locomo benchmark (Maharana et al., 2024) and Long-MT-Bench+ (Pan et al., 2025).

We evaluate a range of approaches, including vanilla LLMs, naive RAG methods, the classic MemoryBank (Zhong et al., 2024), and several recent representative methods such as A-MEM (Xu et al., 2025a), LightMem (Fang et al., 2025), SeCom (Pan et al., 2025), and MemoryOS (Kang et al., 2025). Detailed descriptions of these methods and implementation specifics are provided in Appendix A.

Our experiments focus on scenarios involving locally deployed small-parameter models with short context windows. Unless specified otherwise, we adopt the original prompts and embedding models provided by each method; otherwise, prompts from the Locomo benchmark are used. For all retrieval-based methods, the top-10 most relevant segments are retrieved. To mitigate the impact of context window constraints, chunking and retry mechanisms are applied to certain baselines. Our experiments

[1]Due to space limitations, this section reports only a subset of the experimental results. The complete results of all experiments are provided in Appendix B.

Backbone	Method	Multi-Hop		Temporal		Open Domain		Single-Hop		Token Cost (in thousands)
		F1	BLEU	F1	BLEU	F1	BLEU	F1	BLEU	
Qwen3-4B (Window: 8192)	Only LLM	4.82	3.47	0.64	0.33	1.89	1.12	10.23	8.49	8,164.16
	Naive RAG	1.74	1.09	1.35	0.71	2.63	1.73	2.73	1.33	-
	A-MEM	20.18	13.22	27.79	18.68	3.96	3.85	27.38	23.74	8,395.01
	+MemDecider	24.99*	17.34*	**35.03***	**23.64***	**8.22***	5.56*	**41.06***	**36.78***	6,578.41
	LightMem	5.24	3.83	4.15	2.61	3.84	2.78	11.18	9.51	1,102.92
	+MemDecider	5.00	3.51	4.05	2.65*	2.89	2.01	11.55*	9.67*	705.96
	SeCom	8.60	5.83	11.97	8.96	4.57	3.65	20.54	17.47	1,094.24
	+MemDecider	8.60	5.84*	12.02*	9.04*	4.55	3.64	20.63*	17.55*	1,093.59
	MemoryOS	14.82	11.61	24.84	19.38	5.77	5.09	31.61	28.44	7,019.78
	+MemDecider	15.45*	11.53	24.95*	19.86*	3.66	2.12	32.86*	29.46*	6,915.63
	MemoryBank	2.17	1.62	1.67	1.01	2.15	1.47	5.20	3.41	1,157.69
	+MemDecider	3.38*	2.60*	2.78*	1.80*	3.13*	2.15*	9.45*	6.51*	1,127.79
	EMA	**31.10**	**23.12**	34.65	23.48	7.42	**6.79**	40.93	35.85	6,581.66
Qwen3-4B (Window: 32K)	Only LLM	10.34	7.56	1.04	0.68	4.73	2.69	18.30	15.47	7,990.30
	Naive RAG	2.92	2.08	10.44	7.67	3.24	1.81	8.51	7.21	-
	A-MEM	21.50	14.21	23.39	14.87	4.93	4.45	26.21	22.06	8,409.09
	+MemDecider	18.78	11.57	26.56*	19.26*	5.41*	4.52*	24.19	20.86	7,662.84
	LightMem	4.76	3.34	4.41	2.87	3.94	2.90	11.26	9.42	1,107.37
	+MemDecider	4.68	3.26	5.11*	3.29*	4.04*	2.95*	12.09*	10.15*	710.36
	SeCom	8.63	5.85	12.37	9.26	4.52	3.61	20.63	17.50	1,093.89
	+MemDecider	8.63	5.86*	12.30	9.21	4.53*	3.61	20.63	17.55*	1,093.42
	MemoryOS	15.80	12.09*	25.19	19.65	4.15	3.41	33.21	29.25	7,098.47
	+MemDecider	15.47	11.86	25.41*	20.22*	4.58*	3.74*	32.90	29.15	6,969.33
	MemoryBank	2.77	2.11	3.39	2.24	3.87	2.54	9.92	6.79	1,238.02
	+MemDecider	3.23*	2.35*	3.79*	2.75*	2.89	1.84	10.28*	7.09*	1,157.69
	EMA	**30.21**	**22.53**	**36.37**	**24.65**	**9.26**	**6.59**	**39.14**	**34.08**	6,554.79

Table 1: Performance on the Locomo dataset. **Bold values** indicate the best results, underlined values denote the second-best results, $values^*$ marked with "*" indicate methods improved by MemDecider. EMA clearly outperforms baselines (two-tailed paired t-test with Bonferroni correction, $p < 0.05$) under the same settings.

cover various question types, including Multi-Hop, Temporal, Open-Domain, and Single-Hop queries. We evaluate all the models five times, and report the average results across all experiments. Implementation details are provided in Appendix A.

4.2 Main Results

Table 1 reports the main results alongside the corresponding token consumption[2]. To quantify the benefits of MemDecider for other frameworks, we also present results obtained by integrating MemDecider into baseline methods. From these results, we draw the following observations:

EMA shows competitive performance. The compared methods demonstrate limited effectiveness under constrained conditions, underscoring the advantages of our approach for edge devices and the importance of minimizing redundant input.

Unfiltered memory degrades performance. Under edge device constraints, both Naive RAG and long-context LLM-based methods (e.g., LightMem, SeCom) suffer from performance drops, indicating reduced stability of these approaches in restricted deployment scenarios.

[2]Since the process of Naive RAG differs from other methods, its token consumption is not included in the calculation.

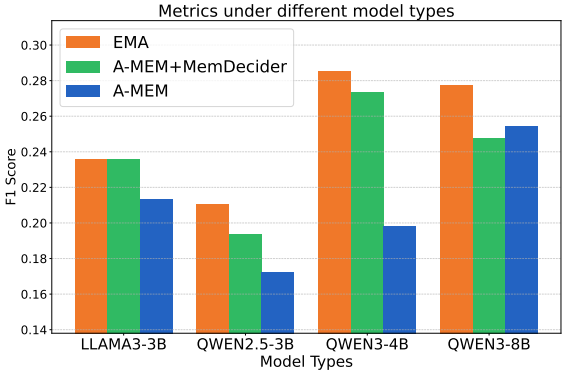

Figure 4: Results across models (average F1), evaluated on the Locomo dataset with 8,192 as windowsize.

MemDecider is an effective plug-and-play module. MemDecider improves most baselines and reduces token consumption. Integrating MemDecider lowers token usage by 12.35% at an 8192 window size and 10.6% at 32768, averaging 11.48%. These results validate its efficacy and generalizability.

Backbone	Method	F1	BLEU	Token(in K)	Backbone	Method	F1	BLEU	Token(in K)
Qwen3-4B (Window: 8192)	Only LLM	13.32	13.36	159.40	Qwen3-4B (Window: 32K)	Only LLM	14.14	13.40	179.51
	Naive RAG	17.98	17.27	-		Naive RAG	17.96	17.24	-
	A-MEM	32.73	19.68	4,021.83		A-MEM	34.91	20.37	5,274.34
	+MemDecider	<u>33.15</u>*	19.53	4,005.19		+MemDecider	<u>35.95</u>*	21.83*	5,245.68
	LightMem	15.08	6.21	261.79		LightMem	15.22	6.26	261.67
	+MemDecider	15.12*	6.41*	257.35		+MemDecider	15.21	6.42*	257.30
	SeCom	26.37	<u>19.82</u>	785.26		SeCom	26.38	19.78	785.24
	+MemDecider	25.91	19.40	783.43		+MemDecider	25.95	19.41	783.40
	MemoryOS	21.52	15.51	2,795.51		MemoryOS	20.13	14.89	2,769.39
	+MemDecider	20.16	14.64	2,658.96		+MemDecider	19.59	14.29	2,597.42
	MemoryBank	10.08	9.94	697.93		MemoryBank	22.78	<u>21.98</u>	950.34
	+MemDecider	10.04	9.72	624.94		+MemDecider	23.21*	**22.30***	946.83
	EMA	**33.79**	**20.11**	3,539.90		EMA	**37.42**	21.84	3,722.96

Table 2: Performance comparison on "Long-MT-Bench+" . **Bold values** indicate the best results, <u>underlined values</u> denote the second-best results, $values^*$ marked with "*" indicate methods improved by MemDecider. EMA clearly outperforms baselines (two-tailed paired t-test with Bonferroni correction, $p < 0.05$) under the same settings.

4.3 Generalizability

We evaluated the performance of MemDecider and EMA across various LLMs. Under a fixed context window of 8,192 tokens, experiments were conducted on multiple open-source models. Figure 4 shows that EMA remains highly competitive, and MemDecider continues to benefit different frameworks. This experiment confirms that our approach is robust across various LLMs.

To further evaluate the generalization capability of EMA, we report experimental results on the Long-MT-Bench+ (Pan et al., 2025) dataset. Compared to Locomo, Long-MT-Bench+ places greater emphasis on extracting and recalling prior information within multi-turn dialogues.

We report the results in Table 2. We observe that EMA consistently achieves optimal performance. Furthermore, MemDecider reduces token consumption across all methods. Notably, MemDecider is applied directly to this dataset without fine-tuning, demonstrating the strong generalization of both EMA and MemDecider.

4.4 Ablation Studies

We conduct ablation studies on each component of our EMA. Unless specified, all experiments in this section utilize the Qwen3-4B model with a context window of 8,192 tokens[3].

First, we systematically analyzed the contribution of each module within EMA, with results summarized in Table 3.

For MemDecider, "All-Accept" denotes committing all inputs to memory, while "Random" represents random memory insertion. These results indicate that both degrade performance; the former introduces noise into subsequent processes, while the latter impairs reasoning. The experimental results here demonstrate the effectiveness of MemDecider, indicating that relevant and important memories are crucial for the overall framework.

Setting	Avg F1	Avg BLEU	Token(in K)
EMA	**28.52**	**22.31**	**6581.65**
All-Accept	28.90	22.10	6994.37
Random	26.65	20.28	6619.52
w/o NER in Caller	28.28	21.08	6419.38
w/o Link in Caller	25.16	19.19	5043.45

Table 3: Module ablation comparing performance and cost across settings.

For Caller, "w/o NER in Caller" uses the raw user input as the query, while "w/o Link in Caller" disables the retrieval of associated EMUs. Both variants result in performance degradation for EMA. The results indicate that our designed mechanisms are beneficial for retrieval.

These ablation experiments demonstrate the indispensable contributions of each individual module in our MemDecider and Caller. It is the synergistic integration of these components with EMUs that establishes a robust abstraction of episodic memory, ultimately forming a cohesive and comprehensive memory-augmented framework.

Second, we analyze the impact of the threshold τ. Figure 5 reveals that performance does not vary monotonically with the threshold; both ex-

[3]For additional experimental details and hyperparameter analysis, please refer to Appendix B and Appendix C.

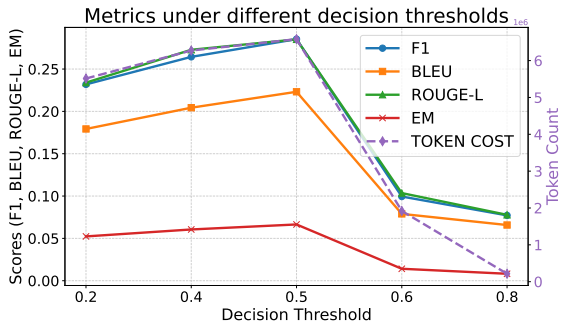

Figure 5: Results under different thresholds. The left axis shows performance,the right axis token cost.

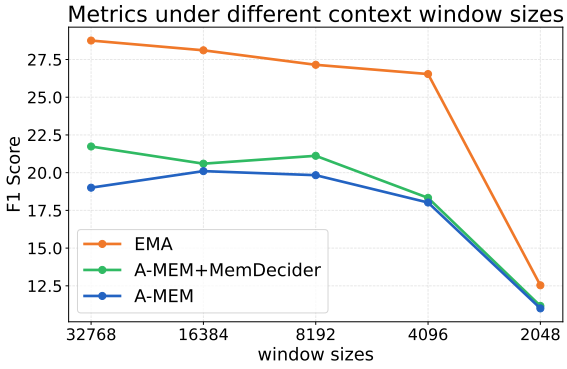

Figure 6: Results under different context window lengths (average F1). The minimum window size is set to 2048, as smaller values lead to method failure.

cessively high and low values lead to performance degradation[4]. This observation supports our core premise that memory content must be selectively filtered: redundant information introduces noise, while insufficient information constrains effective reasoning. Furthermore, this result reflects a trade-off between performance and efficiency, suggesting that the choice should be tailored to practical requirements. We found that setting $\tau = 0.4$ achieves a robust balance between accuracy and efficiency. We provide further discussion in Appendix C.

Third, we evaluate the impact of different context window lengths, as illustrated in Figure 6. Although overall performance typically declines as the window size decreases, MemDecider improves the performance of other methods. Furthermore, EMA demonstrates superior robustness under restricted window lengths, validating the conciseness and effectiveness of the EMU representation.

[4] $\tau \in [0.3, 0.5]$ is suitable for the information-intensive Locomo dataset; the optimal value depends on the scenario.

4.5 Cross-Baseline Gain Analysis

We observe that MemDecider exhibits varying degrees of improvement over different baselines, and we provide a brief analysis in this section. The differences in performance gains across baselines primarily stem from the inherent architectural characteristics of each model and their sensitivity to noise. A more detailed analysis is provided below.

- **For A-MEM and MemoryOS.** These methods incorporate sophisticated memory management mechanisms, including specialized update, linking, and query regeneration modules. By filtering redundant information and providing high-precision memory segments, MemDecider effectively reduces noisy inputs. This enables their complex architectures to operate more effectively, resulting in substantial gains in both performance and efficiency.
- **For LightMem and SeCom.** These frameworks already include internal data processing stages such as compression and noise reduction. Therefore, the marginal benefit of MemDecider is relatively limited due to functional overlap. Nevertheless, it still consistently reduces computational cost.
- **For MemoryBank.** As an earlier exploratory model, MemoryBank often struggles in high-load scenarios such as long-context dialogue. MemDecider mitigates its processing overhead by pre-selecting critical information, thereby improving robustness and performance on challenging datasets.

In summary, although the magnitude of improvement varies across baselines depending on their existing capabilities, the MemDecider consistently improves model performance while reducing overall computational cost. Therefore, MemDecider can be regarded as a general and effective module.

5 Related Work

Multi-turn dialogue serves as a critical context for demonstrating the utility of AI Memory (Wu et al., 2025). Existing AI Memory approaches can be categorized into three classes based on their implementation: RAG-style methods, fine-tuning-based methods, and framework-based methods.

RAG-style methods. It represents some of the early efforts in this field, aiming to adapt RAG workflows to AI Memory scenarios and address the limitations of large language models (LLMs) in

maintaining context and historical memory during long-term interactions (Zhong et al., 2024; Packer et al., 2023; Gutierrez et al., 2024).

Fine-tuning-based methods. These kinds of methods focus on the model itself, attempting to endow LLMs with intrinsic memory capabilities. For instance, MemoryLLM incorporates a fixed-size memory pool with self-updating mechanisms directly into the Transformer architecture (Wang et al., 2024), while other approaches fine-tune models using external inputs augmented with specialized tokens (Yu et al., 2025; Jin et al., 2025). These methods enhance the capabilities of LLMs themselves, enabling improved performance in AI Memory scenarios.

Framework-based methods. These are popular strategies for AI Memory due to their scalability and adaptability. These methods typically introduce a dedicated module or a complete framework that leverages prompt engineering and external memory to enhance AI Memory. For instance, A-MEM (Xu et al., 2025a) combines agentic memory, dynamic indexing, and linking mechanisms to construct a memory network. SeCom (Pan et al., 2025) builds a segment-level memory store and employs compression-based denoising techniques to improve retrieval accuracy. FraCom (Ke et al., 2025) focuses on using LLMs to parse dialogue into propositions for integration and filtering. Our approach also falls into this category, but we construct a complete framework through EMUs. More importantly, we introduce MemDecider as a filtering module to evaluate and select important information.

Current approaches typically omit importance assessment prior to processing input text, delegating all filtering to the LLM and its prompt engineering. This results in the inclusion of irrelevant content in downstream workflows, reducing overall efficiency and increasing resource consumption. To the best of our knowledge, no existing work focuses on text screening before the formal pipeline begins. Inspired by Episodic Memory in human cognition, we aim to construct a lightweight decision module capable of operating at the early stages of the memory process,and goal is for this module to be universal, low-cost, and low-latency. Based on this idea, we abstract the context into EMUs and build MemDecider along with its associated framework, EMA, accordingly.

6 Conclusion

This work highlights that existing research on AI Memory has yet to explore filtering mechanisms based on text importance. To address this, we introduce MemDecider, a lightweight front-end module designed to evaluate information significance and prune redundancy, thereby suppressing noise propagation and computational overhead. Inspired by human episodic memory, we further develop EMA, a comprehensive framework centered on EMUs and MemDecider. Experimental results on two widely-used benchmarks demonstrate that our approach maintains competitive performance while significantly reducing latency and token consumption. As a plug-and-play module, MemDecider can be seamlessly integrated into diverse memory architectures, improving the performance and efficiency of other methods. In conclusion, inspired by human cognitive science, this research optimizes the decision-making pipeline of LLM memory, offering an efficient and scalable solution for AI Memory scenarios.

Limitations

First, our method does not further leverage additional modalities (e.g., vision) to support memory modeling. Since human memory is inherently multimodal, future work will explore incorporating more modalities to enhance memory. Second, personalized assistants and multi-turn dialogue scenarios inevitably raise user privacy concerns. Currently, we do not explicitly consider privacy protection mechanisms, such as asking user consent before storing or updating memory. Future work will place greater emphasis on user privacy and security. Meanwhile, developing robust metrics for evaluating long-term memory consistency remains an open problem, which future work may further investigate.

Acknowledgments

This work is supported by the National Key Research and Development Program of China (No. 2025YFE0113500), the National Science Fund for Distinguished Young Scholars (No. 62525605), and the National Natural Science Foundation of China (No. 62272401 and No. 62572410).

References

Nelson Cowan. 2012. *Working memory capacity*. Psychology press.

Yuhong Dai, Jianxun Lian, Yitian Huang, Wei Zhang, Mingyang Zhou, Mingqi Wu, Xing Xie, and Hao Liao. 2025. Pretraining context compressor for large language models with embedding-based memory. In *ACL (1)*, pages 28715–28732. Association for Computational Linguistics.

Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on RAG meeting llms: Towards retrieval-augmented large language models. In *KDD*, pages 6491–6501. ACM.

Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, Huajun Chen, and Ningyu Zhang. 2025. Lightmem: Lightweight and efficient memory-augmented generation. *arXiv Preprint*. https://arxiv.org/abs/2510.18866.

Bernal Jimenez Gutierrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. Hipporag: Neurobiologically inspired long-term memory for large language models. In *NeurIPS*.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. 2025. From RAG to memory: Non-parametric continual learning for large language models. In *ICML*.

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, Senjie Jin, Jiejun Tan, Yanbin Yin, Jiongnan Liu, Zeyu Zhang, Zhongxiang Sun, Yutao Zhu, Hao Sun, Boci Peng, and 28 others. 2025. Memory in the age of ai agents. https://arxiv.org/abs/2512.13564.

Mingyu Jin, Weidi Luo, Sitao Cheng, Xinyi Wang, Wenyue Hua, Ruixiang Tang, William Yang Wang, and Yongfeng Zhang. 2025. Disentangling memory and reasoning ability in large language models. In *ACL (1)*, pages 1681–1701.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. 2025. Memory OS of AI agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 25972–25981, Suzhou, China. Association for Computational Linguistics.

Cai Ke, Yiming Du, Bin Liang, Yifan Xiang, Lin Gui, Zhongyang Li, Baojun Wang, Yue Yu, Hui Wang, Kam-Fai Wong, and Ruifeng Xu. 2025. Flexibly utilize memory for long-term conversation via a fragment-then-compose framework. In *EMNLP*, pages 21119–21136.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. 2025. Llms get lost in multi-turn conversation. *arXiv Preprint*. https://arxiv.org/abs/2505.06120.

Jiaqi Li, Mengmeng Wang, Zilong Zheng, and Muhan Zhang. 2024. Loogle: Can long-context language models understand long contexts? In *ACL (1)*, pages 16304–16333.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Sheng-Chieh Lin, Akari Asai, Minghan Li, Barlas Oguz, Jimmy Lin, Yashar Mehdad, Wen-tau Yih, and Xilun Chen. 2023. How to train your dragon: Diverse augmentation towards generalizable dense retrieval. In *EMNLP (Findings)*, pages 6385–6400. Association for Computational Linguistics.

Jiaheng Liu, Dawei Zhu, Zhiqi Bai, Yancheng He, Huanxuan Liao, Haoran Que, Zekun Wang, Chenchen Zhang, Ge Zhang, Jiebin Zhang, Yuanxing Zhang, Zhuo Chen, Hangyu Guo, Shilong Li, Ziqiang Liu, Yong Shan, Yifan Song, Jiayi Tian, Wenhao Wu, and 18 others. 2025. A comprehensive survey on long context language modeling. *arXiv Preprint*. https://arxiv.org/abs/2503.17407.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of LLM agents. In *ACL (1)*, pages 13851–13870. Association for Computational Linguistics.

George A Miller. 1956. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, 63(2):81.

Andrew Y. Ng, Daishi Harada, and Stuart Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, pages 278–287.

Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. 2023. Memgpt: Towards llms as operating systems. *arXiv Preprint*. https://arxiv.org/abs/2310.08560.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Xufang Luo, Hao Cheng, Dongsheng Li, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Jianfeng Gao. 2025. Secom: On memory construction and retrieval for personalized conversational agents. In *ICLR*. OpenReview.net.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2024. Yarn: Efficient context window extension of large language models. In *ICLR*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*, pages 2383–2392.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv Preprint*. https://arxiv.org/abs/1707.06347.

Jianlin Su, Murtadha H. M. Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Zhen Tan, Jun Yan, I-Hung Hsu, Rujun Han, Zifeng Wang, Long T. Le, Yiwen Song, Yanfei Chen, Hamid Palangi, George Lee, Anand Rajan Iyer, Tianlong Chen, Huan Liu, Chen-Yu Lee, and Tomas Pfister. 2025. In prospect and retrospect: Reflective memory management for long-term personalized dialogue agents. In *ACL (1)*, pages 8416–8439.

Endel Tulving. 2002. Episodic memory: From mind to brain. *Annual review of psychology*, 53(1):1–25.

Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li, Xian Li, Bing Yin, Jingbo Shang, and Julian J. McAuley. 2024. MEMORYLLM: towards self-updatable large language models. In *ICML*. OpenReview.net.

Yaxiong Wu, Sheng Liang, Chen Zhang, Yichao Wang, Yongyue Zhang, Huifeng Guo, Ruiming Tang, and Yong Liu. 2025. From human memory to AI memory: A survey on memory mechanisms in the era of llms. *arXiv Preprint*. https://arxiv.org/abs/2504.15965.

Yingsheng Wu, Yuxuan Gu, Xiaocheng Feng, Weihong Zhong, Dongliang Xu, Qing Yang, Hongtao Liu, and Bing Qin. 2024. Extending context window of large language models from a distributional perspective. In *EMNLP*, pages 7288–7301.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025a. A-mem: Agentic memory for llm agents. In *Advances in Neural Information Processing Systems*.

Xinhao Xu, Jiaxin Li, Hui Chen, Zijia Lin, Jungong Han, and Guiguang Ding. 2025b. Extending LLM context window with adaptive grouped positional encoding: A training-free method. In *ACL (1)*, pages 573–587.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and Hao Zhou. 2025. Memagent: Reshaping long-context LLM with multi-conv rl-based memory agent. *arXiv Preprint*. https://arxiv.org/abs/2507.02259.

Ruifeng Yuan, Shichao Sun, Yongqi Li, Zili Wang, Ziqiang Cao, and Wenjie Li. 2025. Personalized large language model assistant with evolving conditional memory. In *COLING*, pages 3764–3777. Association for Computational Linguistics.

Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. 2025. A survey on the memory mechanism of large language model-based agents. *ACM Trans. Inf. Syst.*, 43(6):155:1–155:47.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. Memorybank: Enhancing large language models with long-term memory. In *AAAI*, pages 19724–19731. AAAI Press.

A Implementation Details

This section details the implementation of our study. It covers the benchmarks and baselines, the prompts utilized in EMA.

The experiments were conducted on an Ubuntu 22.04.5 LTS system, equipped with 2 Intel Xeon Silver 4314 CPUs , 8× NVIDIA GeForce RTX 3090 GPUs (24GB VRAM each), CUDA driver version 12.2 (nvcc 11.7), with an x86_64 system architecture.

A.1 BenchMark and Baseline

Our experiments focus on two datasets and compare MemDecider with the following methods. Detailed descriptions are provided below:

- **Locomo** (Maharana et al., 2024) serves as the benchmark for our main experiments. It consists of ten samples (for fairness, we uniformly use five samples that MemDecider has not seen). Each sample contains multiple sessions, and at the session level, there are multi-turn dialogues along with multiple associated questions[5].
- **LongMT+** (Pan et al., 2025), reconstructed from MT-Bench+, evaluates dialogue systems in long-term scenarios. Its expanded, challenging questions cover diverse domains and complex logic, facilitating a comprehensive assessment of historical information understanding and utilization[6].
- **Only LLM and Naive RAG.** These two approaches are relatively simple and follow the Locomo settings for such methods. For Only LLM, all dialogues within each session are provided as context, and questions are asked sequentially until the session concludes. For Naive RAG, session-level dialogue embeddings are stored, and retrieval is performed during question answering; the retriever used is Dragon Plus (Lin et al., 2023).

[5]It is licensed under CC BY-NC 4.0.

[6]It is licensed under the MIT License.

- **A-MEM** (Xu et al., 2025a) models user interactions as atomic memory notes and uses LLMs to dynamically link and evolve memories into a self-organizing memory network.
- **LightMem** (Fang et al., 2025) employs a three-stage lightweight memory pipeline with pre-compression, short-term aggregation, and offline long-term updates inspired by human memory models.
- **SeCom** (Pan et al., 2025) builds paragraph-level memory units via topic segmentation and applies prompt-based denoising to improve contextual relevance in long-term retrieval.
- **MemoryOS** (Kang et al., 2025) adopts a hierarchical memory architecture with dedicated modules for storage, updating, retrieval, and generation to support dynamic semantic memory management.
- **MemoryBank** (Zhong et al., 2024) introduces an explicit external memory framework that hierarchically stores dialogues, events, and user profiles with selective recall and forgetting.

A.2 Prompt

A.2.1 Default prompt for QA answers

The prompt in Table 4 is used to guides the model to identify specific information in the provided context and respond with a concise, short phrase.

Section	Content
1. Prompt Text	Based on the context: `{context}`, write an answer in the form of a short phrase for the following question. Answer with exact words from the context whenever possible.
2. Input	Question: `{question}`
3. Output	Short answer:

Table 4: Prompt of QA answers.

A.2.2 Construction of EMUs

The prompt in Table 5 is used to guide LLMs in constructing EMUs from raw data. It transforms unstructured text into a JSON object containing a summary of trigger events, semantic tags, Salience(referred to as importance score in the paper), and confidence.

A.2.3 Link and Update

The prompt shown in Table 6 is used to determine the relationships between a new EMU and existing EMUs and returns a JSON-formatted output to guide subsequent processing.

Section	Content
1. Prompt Text	Analyze the following memory content and extract structured metadata. Please extract the following information and format it as a JSON object: 1. trigger_event, 2. context_tags, 3. salience (0.0-1.0), 4. confidence (0.0-1.0).
2. Input	Content: `{content}`
3. Output	JSON Format: `{{"trigger_event": "...", "context_tags": [...], "salience": 0.0, "confidence": 0.0}}`

Table 5: Prompt of Construction.

Section	Content
1. Prompt Text	You are an AI memory update assistant and manager responsible for maintaining a memory repository. Please analyze the newly input content (including its trigger event, context tags, salience, and confidence) together with the most relevant memories retrieved from the memory base, and make decisions about how the memory should be updated. The input JSON fields provided to you include: Trigger Event: `{trigger_event}`, Content: `{content}`, Context Tags: `{context_tags}`, Salience: `{salience}`, Confidence: `{confidence}`. The most relevant memories retrieved by the retriever are as follows: `{memories}`. Based on the above information, please determine: 1. Relationships; 2. Update necessity; 3. Specific actions; 4. Ensure output length matches `{neighbor_number}`.
2. Input	Structured metadata of new input and a list of retrieved neighboring memories.
3. Output	JSON : `should_evolve`,`actions`, `suggested_connections`, `tags_to_update`, `new_context_neighborhood`, and `new_tags_neighborhood`.

Table 6: EMU linking and updating.

B Additional Experiments

In this section, we provide a more detailed report of the experimental results. This includes in-depth analyses of experiments across different models

Model	Method	Multi-Hop			Temporal		
		F1	BLEU-1	ROUGE-L	F1	BLEU-1	ROUGE-L
LLAMA3-3B	A-MEM	19.58	13.46	16.36	28.23	20.18	28.50
	A-MEM+MemDecider	20.25	13.57	17.90	30.53	23.38	30.55
	EMA	23.72	15.63	21.47	30.56	23.94	30.62
QWEN3-8B	A-MEM	14.93	12.71	17.28	39.70	28.39	39.70
	A-MEM+MemDecider	19.72	13.36	17.31	42.06	32.02	42.06
	EMA	23.41	16.96	23.08	36.52	25.61	36.57
QWEN2.5-3B	A-MEM	14.95	11.53	14.54	23.56	17.50	24.06
	A-MEM+MemDecider	19.42	13.08	19.12	23.00	19.17	22.98
	EMA	20.64	14.51	20.61	25.77	20.84	27.25

Table 7: Comparison of Multi-Hop and Temporal metrics across different models and methods (Averaged).

Model	Method	Open-Domain			Single-Hop		
		F1	BLEU-1	ROUGE-L	F1	BLEU-1	ROUGE-L
LLAMA3-3B	A-MEM	7.18	6.21	7.26	27.81	22.86	27.78
	A-MEM+MemDecider	7.80	6.33	8.45	30.20	24.93	30.50
	EMA	6.91	5.36	6.95	33.05	27.37	33.42
QWEN3-8B	A-MEM	11.35	11.09	14.37	35.67	33.18	39.50
	A-MEM+MemDecider	14.07	13.92	17.86	23.27	20.35	24.90
	EMA	12.98	13.12	15.00	32.66	28.56	34.51
QWEN2.5-3B	A-MEM	3.31	5.68	6.67	24.42	19.97	24.89
	A-MEM+MemDecider	4.40	4.44	5.78	26.77	22.43	27.73
	EMA	4.50	4.24	6.53	30.14	25.48	31.27

Table 8: Comparison of Open-Domain and Single-Hop metrics across different models and methods (Averaged).

Method	2048		4096		8192		16384		32768	
	F1	BLEU	F1	BLEU	F1	BLEU	F1	BLEU	F1	BLEU
A-MEM	11.00	7.92	18.02	13.59	19.83	14.87	20.10	14.50	19.00	13.92
A-MEM+MemDecider	11.17	7.48	18.33	13.53	21.12	15.16	20.59	14.53	21.73	14.05
EMA	12.54	19.47	26.54	20.47	27.15	17.47	28.11	22.16	28.76	21.97

Table 9: Performance comparison (F1 and BLEU) across different context window sizes.

and input lengths, as well as architecture-level ablation studies of the proposed method to assess the contribution of each component to overall performance. These experiments allow for a comprehensive understanding of the method's behavior under various conditions and validate the effectiveness of the design choices.

B.1 Detailed Experimental Results

B.1.1 Results across Models

In this section, we report detailed experimental results of EMA and A-MEM across different models, covering various settings (Multi-Hop, Temporal, Open-Domain, and Single-Hop) and parameter scales. As shown in Table 7 and Table 8, the results indicate that EMA consistently achieves stable and superior performance under all conditions. Additionally, A-MEM can also benefit when integrated within MemDecider.

B.1.2 Results in different lengths

This section reports the detailed values of the context window. Next, we conduct comparative experiments to evaluate the impact of different context window sizes. The experiments start with a context window of 32,768 tokens, the default setting for Qwen3-4B. We then progressively reduce the window size, as shown in Table 9. The results indicate that the performance of all methods remains largely stable until the window is reduced to 8,192 tokens, suggesting that the models can maintain

Method	Multi-Hop		Temporal		Open Domain		Single-Hop		Average		Token Cost
	F1	BLEU	F1	BLEU	F1	BLEU	F1	BLEU	F1	BLEU	(in thousands)
All-Accept	30.10	21.86	37.03	25.96	6.99	4.35	41.50	36.24	28.90	22.10	6994.37
Random	30.73	21.56	31.58	21.07	6.44	5.37	37.86	33.10	26.65	20.28	6619.52
w/o NER in Caller	29.83	20.44	34.47	23.61	7.69	4.18	41.13	36.10	28.28	21.08	6419.38
w/o Link in Caller	24.62	17.75	34.94	23.70	5.92	4.85	35.14	30.45	25.16	19.19	5043.45

Table 10: Ablation study on different modules.

Method	Category 1	Category 2	Category 3	Category 4
both	0.0448	0.0717	0.0516	0.0779
date	0.0451	0.0467	0.1237	0.0851
number	0.0460	0.0737	0.0540	0.0767
EMA	0.3021	0.3637	0.0926	0.3914

Table 11: Performance comparison between MemDecider and heuristic memory selection strategies across different categories.

	Locomo	LongMTplus
Dialogues (Topic)	5	11
Dialogues (Sessions)	128	54
Avg Turns per Session	21.56	26.67
Total Questions	999	288

Table 12: Statistics of the datasets.

stability under relatively large contexts. However, further reductions lead to significant performance drops, and at 2,048 tokens, almost all methods fail to function properly. Notably, EMA consistently maintains strong performance(when the context window exceeds 2048 tokens), demonstrating the robustness and adaptability of the proposed method across different context window sizes.

B.2 Ablation Experiment Details

In this section, we report detailed results of the ablation studies, covering specific metrics under various experimental settings to compare different modules and assess their importance within the overall method. As shown in Table 10, when the system accepts all memories, overall performance improves, but at the cost of substantially higher token consumption. In contrast, removing other functional modules leads to performance degradation to varying degrees. In particular, the Random memory setting and the variant without EMU retrieval in the Caller (w/o Link in Caller) result in the most pronounced performance drops, indicating that effective memory selection and EMU linking mechanisms are critical to the framework.

In addition, simpler heuristics (e.g., only storing user utterances containing dates or numbers) may improve computational efficiency. However, as shown in Table 11, these rule-based strategies perform inconsistently across different categories and can miss important semantic signals that are not explicitly tied to numerical cues. In contrast, MemDecider consistently achieves substantially better performance across all categories, demonstrating the effectiveness of learning-based fused feature modeling for importance estimation. To further validate our design, we conduct an ablation study comparing our method with these heuristic baselines (i.e., “date”, “number”, and “both”, where “both” denotes utterances containing either dates or numerical values). The results are shown in Table 11.

B.3 Dataset

We adopt a consistent evaluation protocol: Locomo is evaluated on a 50% subset unseen by MemDecider, while Long-MT-Bench+ uses the full dataset. For clarity, we provide a summary of dataset statistics in Table 12, including the number of dialogues and average turns for each split. The exact data splits are also available in the `dataset` directory of our repository.

C Hyperparameter Analysis

This section reports on hyperparameters, including experiments on threshold selection and analysis of the reward function’s weight coefficients.

Threshold	Multi-Hop		Temporal		Open Domain		Single-Hop		Token Cost (in thousands)
	F1	BLEU	F1	BLEU	F1	BLEU	F1	BLEU	
0.2	22.75	16.55	30.48	21.02	7.51	5.92	32.01	28.20	5513.84
0.4	27.10	18.86	34.50	23.68	6.59	5.89	37.56	33.31	6272.98
0.5	31.10	23.13	34.66	23.48	7.43	6.79	40.93	35.85	6581.66
0.6	10.04	7.59	10.01	7.09	7.61	6.68	12.11	10.22	1919.86
0.8	5.71	5.50	4.33	3.70	10.69	9.00	10.15	8.11	215.08

Table 13: Performance with different thresholds.

Weights				Multi-Hop		Temporal		Open-Domain		Single-Hop		Average		Token (in K)
w_{f1}	w_{rou}	w_{bleu}	w_{em}	F1	BLEU	F1	BLEU	F1	BLEU	F1	BLEU	F1	BLEU	
0.2	0.1	0.4	0.3	29.1	20.9	37.8	25.3	10.5	9.0	41.6	36.8	29.8	23.0	6,972.62
0.3	0.4	0.2	0.1	31.2	22.9	36.4	25.7	10.1	7.1	41.3	36.2	29.8	23.0	6,932.28
0.2	0.1	0.5	0.2	31.1	22.7	35.4	23.8	7.0	5.4	42.2	37.4	28.9	22.3	6,965.18
0.2	0.1	0.5	0.2	31.3	22.6	36.7	24.3	5.6	4.2	42.0	37.0	28.9	22.0	6,962.60
0.2	0.5	0.1	0.1	28.4	19.8	37.9	27.1	5.5	3.9	42.3	37.3	28.5	22.0	6,983.16
0.5	**0.2**	**0.2**	**0.1**	**31.1**	**23.1**	**34.7**	**23.5**	**7.4**	**6.8**	**40.9**	**35.9**	**28.5**	**22.3**	**6,581.65**
0.4	0.3	0.2	0.1	31.2	22.5	36.0	25.1	6.3	4.9	39.5	34.5	28.2	21.8	6,970.75
0.4	0.3	0.2	0.1	29.5	21.8	34.9	23.9	5.0	4.7	43.3	38.5	28.1	22.3	6,954.55
0.2	0.1	0.4	0.4	27.1	19.5	33.6	23.5	5.5	4.2	41.4	36.7	26.9	21.0	6,949.55

Table 14: Hyperparameter study on weight configurations. The four weights control F1, ROUGE-L, BLEU-1, and EM, respectively. Evaluation and metrics are conducted on the validation set.

C.1 Decision thresholds

In this section, we report ablation results by varying the decision threshold, which determines whether newly observed information should be written into memory. A higher threshold imposes a stricter criterion for memory writing. Table 13 summarizes the results under different threshold settings. When the threshold is set too high, very little information is stored in memory, leading to insufficient historical context for the model to exploit and a clear degradation in overall performance. Correspondingly, token consumption is also reduced, indicating that effective retrieval and response processes are rarely triggered.

Conversely, when the threshold is set too low, excessively low-quality or noisy memories are written into the system, which also results in performance degradation. We attribute this to the fact that, at a threshold of 0.2, the abundance of noisy memories hinders effective Link and Update operations, and noise accumulation further weakens the model's multi-hop reasoning capability. Although token consumption under this setting is lower than that with thresholds of 0.4 and 0.5, the associated performance loss outweighs the efficiency gains, contradicting our objective of reducing cost while preserving performance.

Overall, considering the trade-off between performance and token consumption, we find that thresholds of 0.4 or 0.5 provide a more balanced setting. In subsequent experiments, we therefore adopt 0.5 as the default decision threshold.

C.2 Weight coefficients in the reward function

This section presents experiments on the weight coefficients in our reward function.

The reward R is defined as a composite signal:

$$R_{qa} = w_1 \cdot (\text{F1}) + w_2 \cdot (\text{ROUGE-L}) + w_3 \cdot (\text{BLEU-1}) + w_4 \cdot (\text{EM}),$$

$$R_{total} = \alpha \cdot R_{qa} + \beta \cdot \sum_{t=1}^{T} r_{step,t}.$$

Here, we further analyze the effect of different values of w_k, as shown in the Table 14. Except for a few cases, performance variance remains marginal across different values of w_k. Given this insignificant gap, and to prioritize the optimization of latency and token costs, we ultimately selected the weights $w_1 = 0.5$, $w_2 = 0.2$, $w_3 = 0.2$, and $w_4 = 0.1$ for the reward function during training (**as highlighted in bold in Table 14**, corresponding to F1, ROUGE-L, BLEU-1, and EM, respectively).